ARRL AMATEUR RADIO

7TH COMPUTER NETWORKING CONFERENCE

COLUMBIA, MARYLAND

OCTOBER 1, 1988

7TH COMPUTER NETWORKING CONFERENCE

Coordinators:

Tom Clark, W3IWI
Andre Kesteloot, N4ICK
Dave Tillman, WA4GUD
Jack Colson, W3TMZ
Paul L. Rinaldo, W4RI

Hosts:

Johns Hopkins Applied Physics Laboratory Amateur Radio Club
Tucson Amateur Packet Radio (TAPR)
Amateur Radio Research and Development Corporation (AMRAD)
Radio Amateur Satellite Corporation **(AMSAT)**
American Radio Relay League (ARRL)

American Radio Relay League
Newington, CT USA 06111

$12.00 in USA

ISBN: o-87259-213-8

First Edition

Foreword

The American Radio Relay League is pleased to publish papers for this Seventh ARRL Amateur Radio Computer Networking Conference near Washington, DC, October 1, 1988.

As in past conferences, the papers represent increasing sophistication in packet-radio techniques developed by radio amateurs throughout the world. High-speed modems, protocols and packet satellites dominate this Seventh conference. It is ironic that the Federal Communications Commission has decided to reallocate the 220-222 MHz band to the land mobile service at a time when amateur 56-k/bit packet-radio modems are finally ready to make an impact in this band.

Estimates vary, but most observers place the number of packet-radio TNCs between 50,000 and 80,000 worldwide, as of mid-1988. It is clear that amateur packet radio is still growing at a remarkable rate. In the past year, packet radio has come into its own as a primary handling-traffic medium and is now well poised to provide disaster-communication service in numerous countries.

David Sumner, K1ZZ
Executive Vice President

Newington, Connecticut
September, 1988

Trademarks and service marks appearing in these papers are:

AMSAT is a registered trademark of The Radio Amateur Satellite Corporation
ANSI is a registered trademark of the American National Standards Institute
Apple is a trademark of Apple Computer Inc.
AutoCAD is a registered trademark of Autodesk Inc.
C-64 is a trademark of Commodore Business Machines Inc.
Diconex is a trade name of Kodak
Flexar is a trademark of Motorola
Heliax is a trademark of the Andrews Corporation
IBM is a registered trademark of International Business Machines
Macintosh is a trademark of Apple Computer Inc.
MAXAR is a trademark of Motorola
Micon is a trademark of Motorola
MS-DOS is a registered trademark of Microsoft Corporation
NET/ROM is a trademark of Software 2000 Inc.
OrCAD is a trademark of OrCAD Systems Corporation
Turbo Database Toolbox is a trademark of Borland International Inc.
Turbo Pascal is a trademark of Borland International Inc.
TYMNET is a trademark of Tymshare
UNIX is a registered trademark of AT&T
Z80 is a registered trademark of Zilog Inc.

Contents

A DUPLEX PACKET RADIO REPEATER APPROACH TO LAYER ONE EFFICIENCY

PART TWO

Scott Avent, N6BGW   Robert Finch, N6CXB
c/o South Coast Radio Relay
4446 Rosebank Drive
La Canada, California 91011-1928

BACKGROUND

Last year the authors presented the first part of this paper at the 6th conference by presenting related material NOT contained within the text of the published paper itself. The intent of the verbal presentation was to spur interest in reading the published paper itself. This presentation contained a simple mathematical examination of simplex versus duplex repeater approaches to local area network implementation. In contrast the paper contained the design goals and basic equipment approach we implemented in our duplex repeater environment here in Southern California.

THE PURPOSE

The purpose of our verbal presentation was to startle the audience into accepting duplex packet repeaters' INHERENT superior performance profile. We supported this position with 'hard numbers' presented in chart form. We then hoped this audience 'realization' would lead to examination of our written paper.

Many conference attendees asked us to consider publishing the verbal presentation material. We therefore are including these charts for publication this year. Some explanation is required to accompany these charts.

THE CHARTS

Chart number one is the visual representation of the network 'topography' associated with simplex store and forward repeating's "HIDDEN TERMINAL" effect. Of course it is a problem because it leads to channel capacity limitation due to collision potential. This is covered in depth as basic packet primer material in many papers and networking texts. (Does anyone in ham packet not know about this?) But it does serve a useful purpose in logically leading into the next chart.

Chart number two is a less well known (at least by name) simplex digital effect called **EXPOSED TERMINAL? While the newer versions of layer two and especially layer three, which have become prevalent within the last year, have a tendency to diminish the effects of this simplex related problem, they by no means eliminate it to the extent that duplex does.

Chart number three examines why in some visual detail. Topographic R.F. barriers are of no concern if a properly coordinated AND balanced duplex repeater is used. As long as a user can 'hear and **copy'** the duplex repeater's output these barriers and users' relational proximity to them can do no harm.

Chart number four explains how this lack of susceptibility to hidden terminal effects relates to improved performance for a duplex packet repeater. On the top left half of the page the transmission timing of the simplex LAN (local area network) is totaled and the percentage of this time which is exposed to possible collisions is determined. This assumes two simple constructs. First a packet length of three seconds, **which** the authors feel is a valid average that occurs, at least here in southern California, on **LAN's.** The second assumption is that all user transmissions and delay intervals are time periods in which collisions may take place. On the right half the same **user's** packet is examined on a duplex based LAN. But this time since there are no hidden terminals, the **user's** packet transmission interval is NOT available for possible collisions, and therefore only the delay intervals are added to achieve a percentage of time exposed to these collisions. Since a duplex repeater doesn't store and forward, there is just one packet transmission interval, and approximately one-half the total time slot of a simplex store and forward relay of the packet.

From here the chart makes a simple comparison of total time figures. With no collisions taking place the duplex channel has a 10% advantage of speed over a simplex channel, when adjusted for bandwidth. But perhaps no collisions is not a real world issue? Well then the chart does examine a heavy loaded channel. The equation for this % of duplex traffic handling improvement = 100 x (1 minus (((1 + E duplex) x T duplex) / ((1 + E simplex) x T simplex))), where E is exposure time to collisions and T is the total packet timing interval. This figure is a whopping 37% EVEN WHEN ADJUSTED FOR BANDWIDTH.

It should also be noted that this chart does not take into account any of the exposed terminal effects. Basically since an exposed terminal is outside of the topographical boundaries of a duplex repeater's LAN, it can be safely assumed to have only detrimental effects for a simplex network. Therefore the 37% figure above is probably conservative in most highly populated areas with the attendant scarcity of resource and therefore geographically **'reused'** and heavily loaded channels.

Chart number five is self explanatory on the right side but requires some guidance on the left. Item number one assumes that the addition of exposed terminal effects can safely double the performance degradation of simplex repeaters and therefore the 68% figure. Items three through seven are explained or are logical extensions of the material covered in the authors' paper from last year. Item number eight assumes that like all good voice repeaters the duplex repeater will have a well balanced receive to transmit coverage, and unlike simplex repeaters isn't plaqued by poorly coordinated channel and user activity.

The final chart, number six, is for reference. It is explained in detail in last year's paper (part one). However when we wrote last year's paper we were using a directly connected rs-232 internetwork access port, Since that time the current four port interconnect has been relocateed to another site and the current repeater configuration is as drawn in this chart.

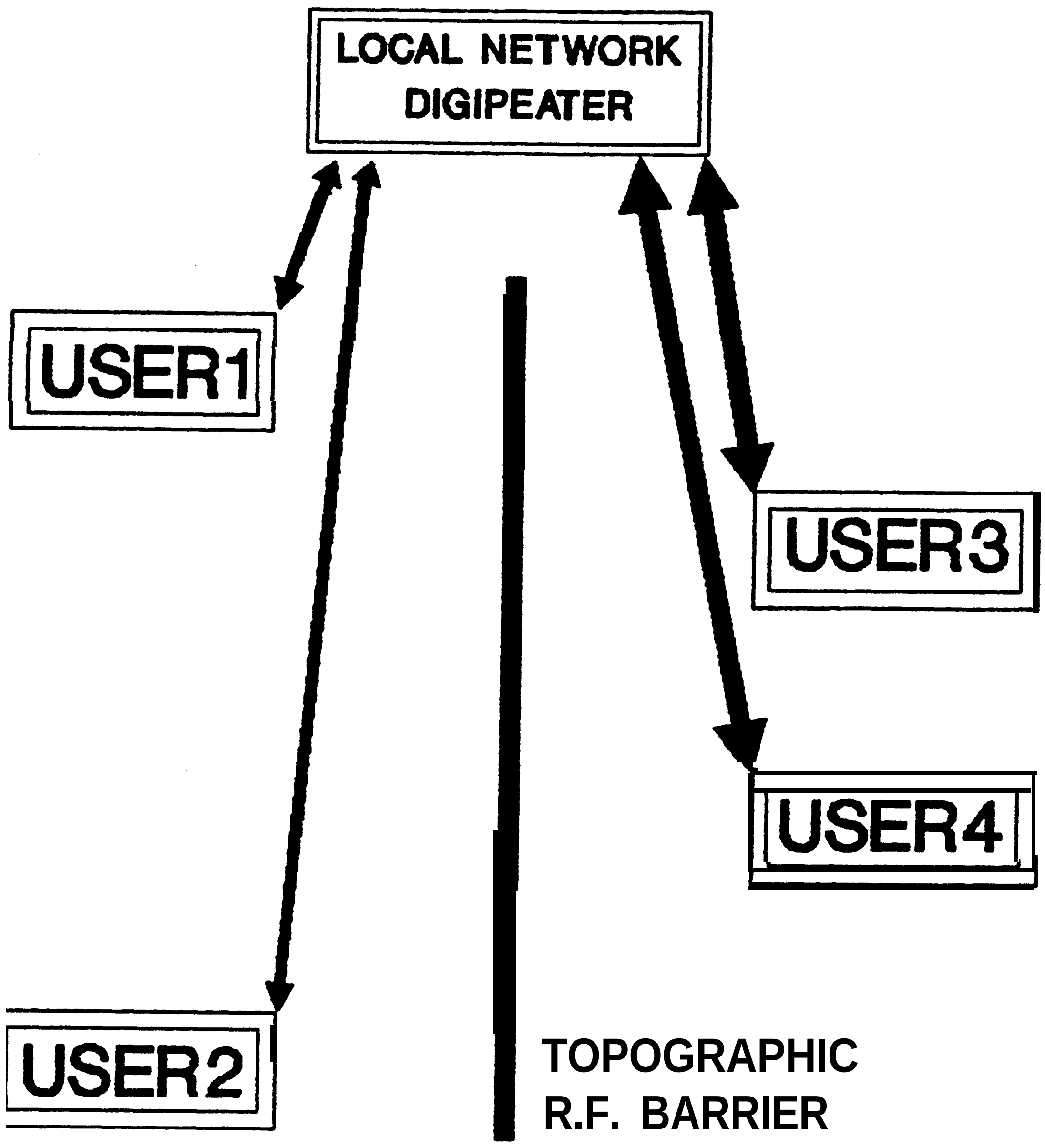
IN CONCLUSION

We hope that these charts will begin to open up thought with the further hope that other parts of the country will begin to have duplex packet LAN's. Since the publication of part one of this paper, the authors have heard of but a scant few duplex repeaters being opened up for packet use. We can think of no better ways, all other things being equal, to significantly improve layer one, than the addition of some cavities, a 202 modem, and some shielding. Failure to provide an efficient layer one network base, is like building a skyscraper upon a foundation of sand.

CHART 1

SIMPLEX PACKET NETWORK

'HIDDEN TERMINAL' EFFECT



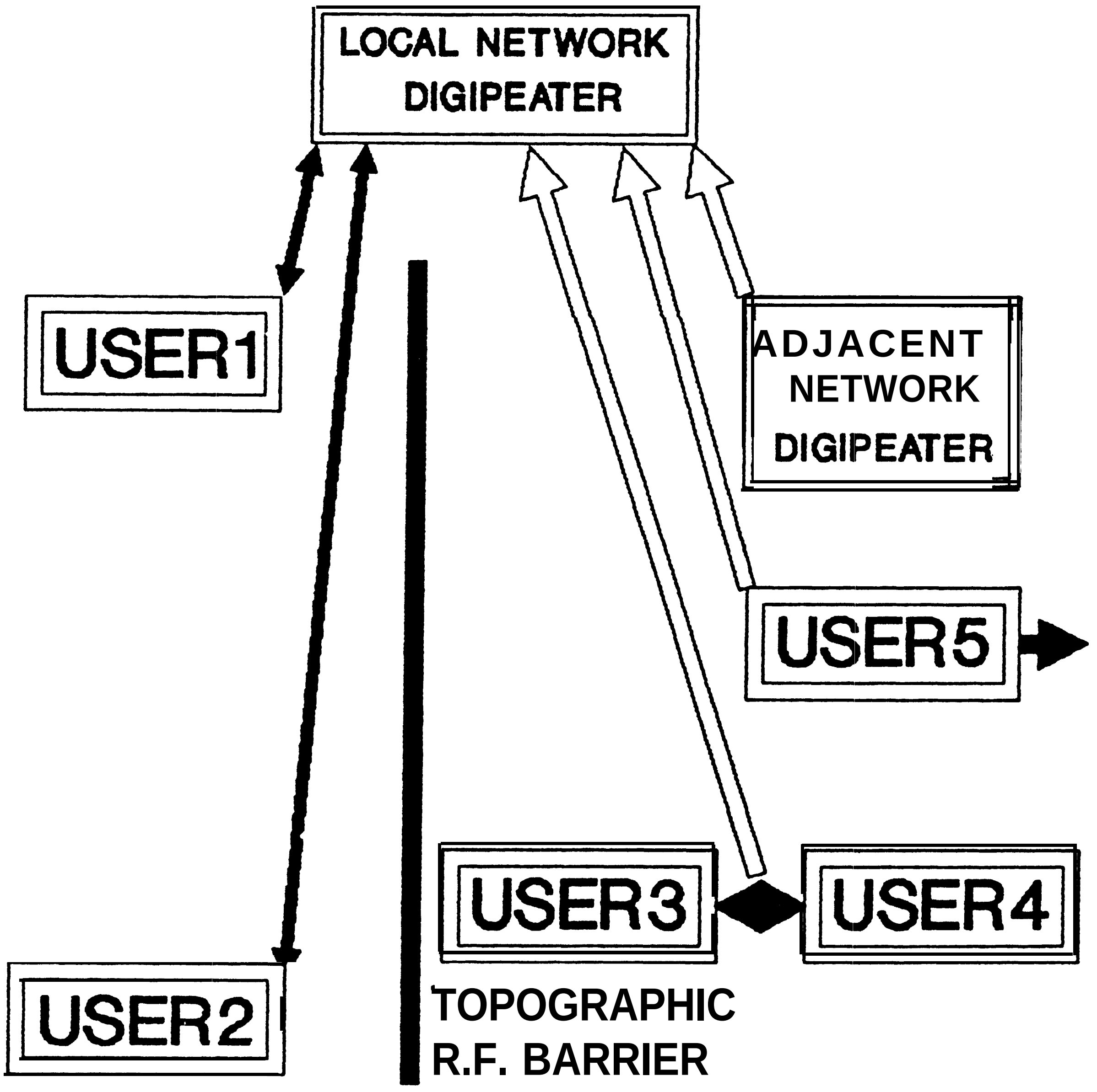
Users cannot copy those on the other side of the topographic barrier directly

User 1 will destroy packets from 3 and 4

users 3&4 will destroy packets from 2.

CHART 2

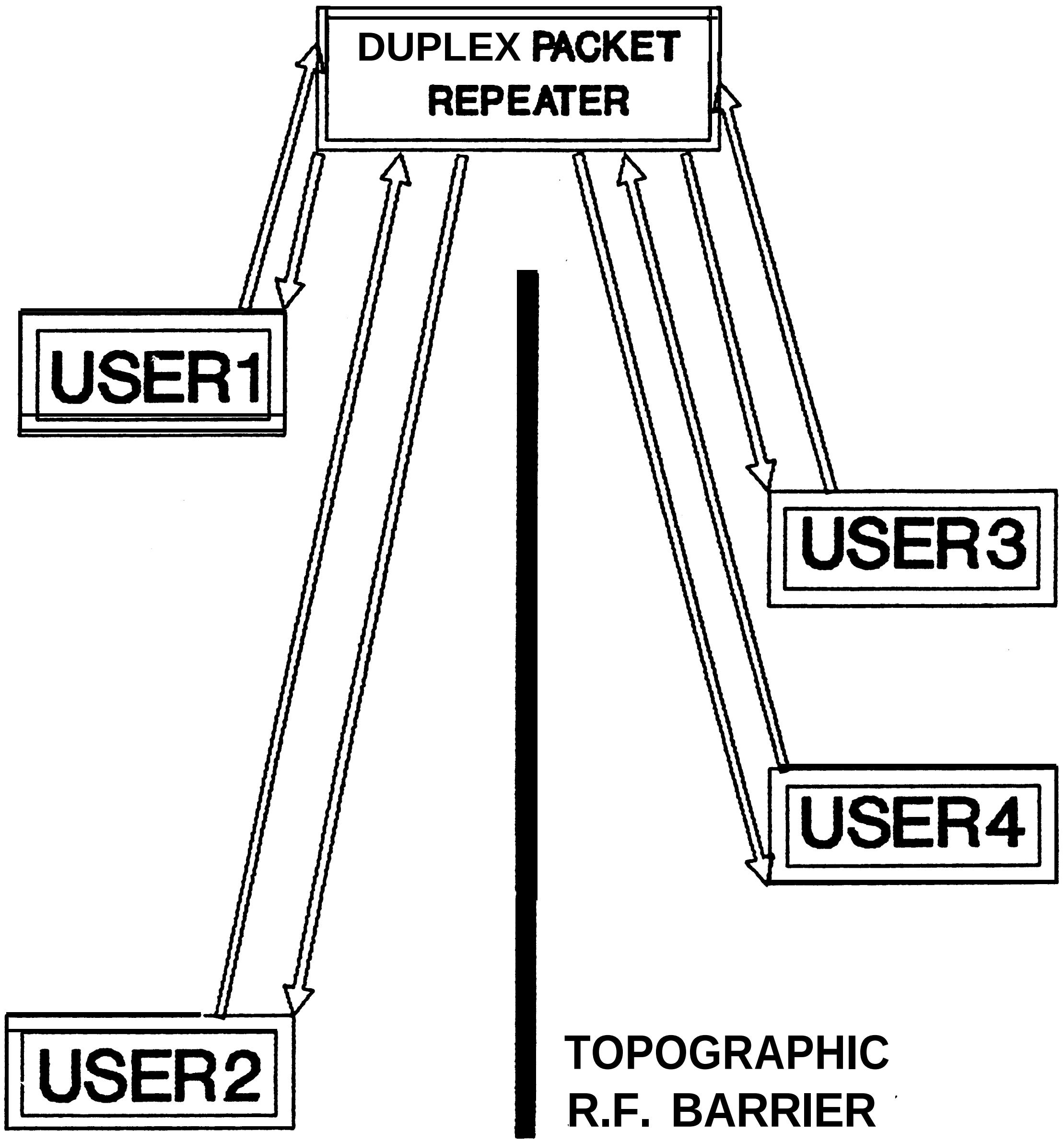
SIMPLEX PACKET NET'WORK
'EXPOSED TERMINAL' EFFECT



Distant signals from adjacent digipeater direct users 3 & 4, and DX'er 5, cause local digipeater to hold off all output transmissions until channel is clear.

CHART 3 --

DUPLEX PACKET NETWORK
COLLISION RESISTANT LAN



Each user can copy all others on the LAN regardless of physical location.

Users transmit on uplink channel and receive on output channel of repeater.

CHART 4

COMPARISON OF DATA TRANSMISSION TIMING AND PROBABILITY OF COLLISIONS OCCURING ON A SIMPLEX AND DUPLEX LAN

SIMPLEX LAN (ONE DIGIPEAT)	TIME (mSec)
User1 keyup dly	10
Digi DCD dly	10
User1 XMIT	3000
Digi keyup dly	10
User2 DCD dly	10
Digi XMIT	3000
User2 T2 timer	1000
User2 keyup dly	10
Digi DCD dly	10
User2 ACK	1000
User1 DCD dly	10
Digi ACK	1000
TOTAL TIME (mSec)	9070

Percentage of time exposed to collisions: 44.9 %

DUPLEX LAN (REPEATER)	TIME (mSec)
User1 keyup dly	10
Rptr DCD dly	20
Rptr keyup dly	0
User2 DCD dly	10
User1 XMIT	3000
User2 keyup dly	10
Rptr DCD dly	20
Rptr keyup dly	0
User1 DCD dly	10
User2 ACK	1000
TOTAL TIME (mSec)	4080

Percentage of time exposed to collisions: 2 %

Transmission speed advantage (apparent to the end user) of a duplex LAN over simplex, assuming that no collisions have occured during the transmission: 55 %

A duplex system requires TWO channels to operate, which would be equivalent to TWO simplex networks simultaneously.
Again, we assume no collisions have occured.
Advantage of Duplex adjusted for RF bandwidth used: 10 %

The real advantage of a duplex LAN comes out when the channel is heavily loaded. Here is an example using the arbitrary collision exposure figures shown above. For pedagogical reasons we assume that retries will occur at a rate approaching the percent exposure to collisions, and that the data will make it through *on* the first *retry.* All stations are assumed to be 'Hidden Terminals' in this model-

Advantage of duplex (apparent to the end user) in a heavily LAN, not adjusted for RF bandwidth used: 68.3 %

Advantage of duplex in a heavily loaded LAN, adjusted for RF bandwidth used: 36.7 %

CHART 5

COMPARISON OF FEATURES BETWEEN LAN TYPES

DUPLEX LAN (repeater)

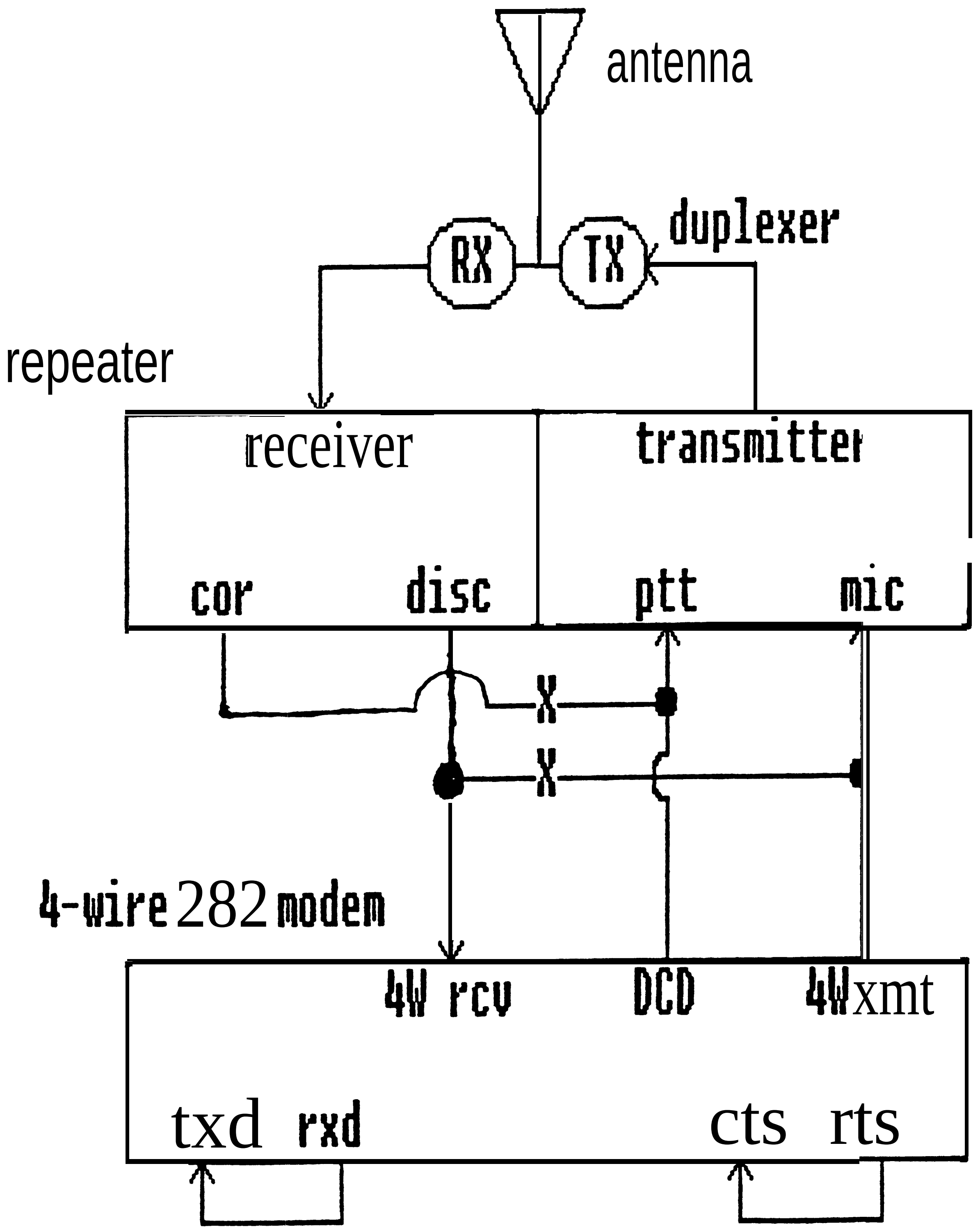
* Up to 68% higher data throughput
* Superior collision resistance
* Can be used with various protocols simultaneously, making it a good test bed for advanced data transmission protocol experiments
* Can support 'Roundtable' nets
* Can be linked to a 'Backbone' net using Dualport or Net/Rom node
* Existing voice repeater can be shared or converted entirely to digital use
* Cash outlay to convert an existing repeater is fairly low (if no network access is desired)
* LAN boundaries are well defined, by repeater coverage area

SIMPLEX LAN (digipeater)

* Simpler and cheaper to construct
* Channel assignments more available
* Easier to network in low use areas
* Requires only half the RF bandwidth of a duplex LAN
* Can be linked to a 'Backbone' net with Dualport or Net/Rom software
* Supported better by manufacturers and software designers

CHART 6

DuplexRepeaterBlockDiagram



FORMAL DESCRIPTION IN ESTELLE OF AX.25

Michel Barbeau, VE2BPM
7429 Casgrain
Montreal, Canada
H2R 1Y4
E-mail: barbeau@iro.umontreal.CA

Abstract
A protocol description must be as precise as possible in order to avoid multiple subjective and incompatible interpretations by the different implementers. Natural language (e.g. english) descriptions are a priori easy to understand but they are in general ambiguous. It has been now recognized that formal techniques give rise to more precise and complete descriptions of software. Estelle is such a formal technique specially designed for communication protocols. This paper investigates the use of Estelle for describing AX.25 a link-level protocol for packet-radio.

1. Introduction

The experience has demonstrated that the clearest thing in most natural language protocol descriptions is their unclearness. This fact has motivated protocol designers to develop various description techniques with theoretical foundations. Among those, methods based on Finite State Machines (FSMs) and elements of programming language are very popular because while being forma? they remain relatively easy to understand by any software practitioner. Estelle is a description technique based on FSMs and a superset of Pascal. One of the interests in Estelle is its acceptance by many protocol designers and implementers. Estelle is published by the international organization for standardization ISO[1].

Good introductions to Estelle can he found in references 2and 3. To save space, we will mention here only the main features of Estelle. A protocol is described in Estelle as a collection of modules. Each module is a state machine capable of having memory, this is termed *Extended* FSM (EFSM). Modules can communicate with each other as well as with the environment of the specified protocol over channels (FIFO queues). Interactions between protocol entities (e.g. frames) are communicated in the channels. Decomposition of a protocol entity into modules is usually functional: a module that defines the transition function of the protocol, a module for mapping frames into interactions with the environment,, etc.

The language of Estelle is based on Pascal with extensions adapted to protocol specification. A transition of a module EFSM is coded using the following constructs. The **from/to-clauses** specify the major state change realized by the transition. The dependence of the transition on an input interaction is expressed using a **when-clause.** A transition without a when clause is said spontaneous. A spontaneous transition can be used to describe internal decisions of the protocol. The condition for firing a transition is expressed in a **provided-clause.** A condition is a boolean expression on input. interaction parameters and or on module context variables. The actions of a transition are put in a Pascal like **begin-end** block Actions can be for example assignments to context variables, calls to procedures or Estelle **output** statements. A complete specification of a simple link-level protocol (the alternating-bit) can be found in references land 2.

The original description of AX.25 packet-radio link-layer protocol is written in english[4]. We intend to demonstrate how a higher degree of formalism can be obtained with Estelle. A short survey of related experiences is given in section 2. Our approach is discussed in section 3. We conclude with section 4.

2. Related Work

AX.25 is an adapted version of ISO's HDLC (or other similar link level protocols such as CCITT's LAPB). Because of its context of use (over HF, VHF or UHF radio channels), AX.25 differs on two main points. First, whereas most link-layer protocols assume a master/slave (DCE/DTE) operation, AX.25 is a symmetrical protocol. The two entities at. the ends of the link are alike, the term DXE is used to refer to both devices. Second, each frame (*U*, *S* as well as *I* frames) always identify both the source and the destination DXEs. Moreover, to allow repeater operation the destination address field may contain a route specification that can be made of up to eight, repeater-addresses.

An at tempt to formalize the procedures of HDLC has been realized by Harangozo[5]. He uses a model incorporating regular grammars and a technique of indexing. Another met hod. using a model closer to Estelle, was used by Bochmann and Chung[6,7]. As in Estelle, they use state machines extended with context variables and Pascal like statements. Their description provided a basis for an implementation[8]. Taylor' presented a structured approach based on the notions of module, FSM and Pascal pseudo-code to describe an implementation of AX.25. His concept, of module interface is close to the concept of module header/body in Estelle. Here we will show the use of the formal description technique Estelle to define the link-level protocol AX.25. A similar experience with Estelle for an application-layer protocol can be found in reference 10.

3. AX.25 Description

A general approach to protocol description is presented in reference 11. The main points that. must be covered are context of the specification, protocol service, internal structure of the protocol, types of exchanged messages and the behavior of the protocol. Those aspects must be defined to a degree necessary to ensure compatibili ty between AX.25 entities but. not further. To save space, a simplified version of AX.25 will be described. It will support. only the I (Information), RR (Receive Ready), SABM (Set Asynchronous Balanced Mode), DISC (Disconnect), DM (Disconnected Mode) and UA (Unnumbered Acknowledgment) frames.

3.1 General Context

AX.25 is to be used as a level 2 protocol of ISO's seven layers reference model,. Its purpose is t 0 provide error free point -to-point communication channels between geographically distributed packet-radio compu ter stat ions. The protocol is designed to operat e over low speed high error rate shared radio channels (physical-layer). More details about the context can be found in reference 4.

3.2 Protocol Service

A protocol provides a given service to its users (e.g. communication channels). A protocol service is specified in terms of the command types (service primitives) available to the user. The service primitives are abstractly defined. Their physical realization in a particular environment (e.g. interrupts, procedure calls, etc.) is left. to the programmers.

AX.25 leaves completely open the interface definition between the protocol and its users. There is no explicit. constraint, on the structure of the messages and their relative ordering at this interface. The protocol defines only the rules for exchanging blocks of data (frames) between AX.25 entities over the physical layer. This is also the case with specifications of HDLC and LAPH. We M-ill t herefore reflect this designer's decision by leaving complete freedom on the service specification to the implementer.

3.3 Structure of the Description

We structure the functions of the protocol into two modules (see figure 1), namely module **Abstract Protocol** (AP) and **Transmitter/Receiver** (TR). The AP module contains the procedures of the link-level protocol but. it makes abstraction of frame encoding in terms of bits and bytes. The functions realized in the TR module are i) encoding/decoding of frames, ii) adding/removing of leading and trailing flags, iii) computing and adding FCS (Frame-Check Sequence) and iv) discarding frames received with incorrect FCS, invalid frames and improperly addressed frames. We will only consider the definition of the AP module.

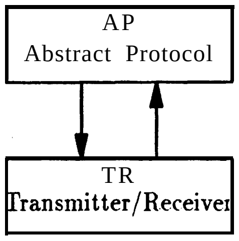


Fig. 1 Structure

3.4 Data Structures

Blocks of data exchanged between AX.25 protocol entities are called frames. There are three basic types of frames: *Unnumbered*, *Supervisory* and *Information*. Each frame consists of a field sequence.

At the level of the AP module we make abstraction of frame encoding in terms of bits and bytes. This task is left to the TR module. The interface between the AP and TR modules is defined by the following channel.

```
channel Abstract_Frames_Channel( User,Provider);
  by User out._frame( frame:frame_type);
  by Provider in_frame( frame:frame_type);
```

The role of *user* will be assigned to the AP module, whereas the TR. module will have the role of *provider.* A Pascal variant record is used to the define the frame data type.

```
frame-kind = (I,S,U);
frame-type = record
  case tag:frame_kind of
    I:( if:I_frame_type);
    S:( sf:S_frame_type);
    U:(uf:U_frame_type);
  end
```

The list of fields making up each frame kind is defined with a record data structure. For example the *I* and *U* frame kinds and their fields are defined as:

```
Address-type = array [1..7] of char;
Addressfield = record
{ repeater operation is not supported }
  destination, source:Address_type;
  end;
 I Control_field = record
   NS.NR:0..7;
   P:boolean;
   end;
 U Control_field = record
   MM:(SABM,DISC,DM,UA);
   P:boolean;
   end;
 PID_field = ( AX25,IP,Addr_res,no_lay_3,esc);
 Info-field = record
   data: array [0..255] of char;
   length :integer;
   end;
 I_frame_type = record
   Address:Address field;
   Control:I Control_field;
   PID:PID_field;
   Info :Info_field;
   end;
 U_frame_type = record
   Address:Address_field;
   Control:U_Control_field;
   end;
```

S frame structure can be similarly defined. The encoding/decoding procedures in the TR modules would be defined based on the information provided in reference 4. While the encoding rules must be strictly respected by the implementer, we generally try to leave as much freedom as possible on the sequence of operations to achieve the desired encoding. It is therefore expected that a large part of the TR module would remain more **or less** informally specified.

3.5 Abstract Protocol

In the AP module we will define actions in response to frames from the peer entity and actions internally initiated. The interaction point with the TR module is named *fc* and the channel used at this point is of type *Abstract_Frames_Channel*. The module header would therefore look like:

```
module Abstract-Protocol
 (T1:integer; MyAddress:Address_type);
 ip
 fc:Abstract-Frames-Channel: end;
```

The Acknowledgement timer T1 value and the local DXE address are specification parameters. The internal behavior of AP is described in the module ***body.*** In the body declaration part are listed major state names under the ***state*** statement.

```
state sl, s2, . . . . s16;
```

This FSM is extended witlh the send and receive variables V(S) and V(R). V(S) contains the number of the next I frame to send. The number of the next expected I frame is kept in V(R). The variables *RemoteAddress* and ***Buffer*** are also defined.

```
var VS, VR:0..7;
 RemoteAddress:Address-type;
 buffer:frame_type;
```

Pascal procedures are defined for building up frames from their parameter values. For example, the procedure *Format_SABM* is defined as:

```
procedure Format-SABM( d,s:Address_type;
    var b:frame_type);
begin
with b do begin
  tag:=U;
  uf.Address.destination:=d; uf.Address.source:=s.
  uf.Control.MM:=SABM;
end; end;
```

The module has the following initialization part It starts in the major state *s1.*

```
initialize
  to sl
begin end;
```

The possible state changes of AP are defined by the transitions. The transition list is divided into five sections (phases): *disconnected*, *link-setup*, *disconnect-request*, *information-transfer* and *waiting-acknowledgement*. The disconnected and link-setup phases are discussed.

A **disconnected** DXE initiates a link setup by transmitting a SABM command to the remote DXE, starting timer T1 and going to the link-setup phase. The Estelle *delay-clause* is used to model the timer Tl. After the module has been in state s2 for T1 time units, the transition with the delay clause (bellow) is enabled. The implementer determines how *RemoteAddress* is obtained.

```
trans
from sl to s2
begin
  RemoteAddress:=...;
  Format-SABM( RemoteAddress,MyAddress,b);
  output fc.out_frame(b);
end;
```

Upon receipt of a SABM command, a disconnected DXE returns a UA response. resets its send and receive variables VR and VS to zero and considers that the link is setup, i.e.. it enters *information-transfer* phase.

```
trans
from sl to s5
when fc .in frame
provided frame.tag= U and
  frame.uf.Control.MM=SABM
begin
  RemoteAdrress:=frame.uf.Address.destination;
  Format_UA(RemoteAddress,MyAddress,b);
  output fc.out_frame(b);
  VR:=0; VS:=O;
end;
```

On receiving a DISC command in the disconnected state, the DXE sends a DM response and remains in the disconnected state.

```
trans
from sl to sl
```

```
when fc.in_frame
provided frame.tag=U and
  frame.uf.Control.MM=DISC
begin
  RemoteAdrress:=frame.uf.Address.destination;
  Format-DM( RemoteAddress,MyAddress,b);
  output fc.out_frame( b);
end;
```

In the disconnected state, the DXE ignores and discards any frame from the remote DXE, except SABMs and DISCs.

```
trans
from s1 to s1
when fc.in_frame
provided not(frame.tag=U and
  (frame.uf.Control.MM=SABM or
  frame.uf.Control.MM=DISC))
begin end;
```

Upon reception of a UA response. in the **link-setup** phase, the local DXE will reset. its send and receive variables VS and VR to zero, stop timer T1 (implicitly realized by the transition) and consider that the link is setup.

```
trans
from s2 to s5
when fc.in_frame
provided frame.tag=U and
  frame.uf.Control.MM=UA
begin
  VR:=0; VS:=0;
end;
```

After the DXE has sent the SABM command, if a **UA** or a DM is not correctly received then timer T1 will run out. The DXE will then resend the SABM and will restart timer T1 (implicit).

```
trans
from s2 to s2
delay (T1)
begin
  Format_SABM( R.emoteAddress,MyAddress,b);
  output fc.out _frame( b);
end;
```

The DXE having sent the SABM command will ignore and discard any frame from the remote DXE, except SABMs, DISCs, or UAs.

```
trans
from s2 to s2
when fc.in_frame
provided not(frame.tag=U and
  (frame.uf.Control.MM=SABM or
  frame.uf.Control.MM=DISC or
  frame.uf.Control.MM=UA))
begin end;
```

In the link-setup phase, the receipt. of a SABM command from the remote DXE will result. in a collision. The local DXE sends a UA response and considers that the link is setup.

```
trans
from s2 to s5
when fc.in_frame
provided frame.tag=U and
  frame.uf.Control.MM=SABM
begin
  VR:=0; VS:=0;
end;
```

On the receipt of a DISC command, the DXE sends a DM response and enters disconnected phase.

```
trans
from s2 to sl
when fc.in_frame
provided frame.tag=U and
  frame.uf.Control.MM=DISC
begin
  Format-DM( RemoteAddress,MyAddress,b);
  output fc.out_frame( b);
end;
```

Would remain to be specified the disconnect-request, information-transfer and waiting- acknowledgement phases. Those phases are described easily in Estelle using the clauses that have been used up to now.

4. Conclusion

We presented the use of Estelle for describing AX.25 a link-level packet-radio protocol. The translation in Estelle was made easier because the informal description contains transition tables. The tables provided a skeleton for the formal specification. The

AP module gives a high level abstract view of AX.25 because many details not relevant to the understanding of the protocol behavior are delegated to the TR module.

Directions for further work would be: i) completion of the description in Estelle of AX.25 and ii) evaluation of other formal description techniques, such as Lotos[12] or SDL[13], for description of AX.25.

References

[1] ISO/TC 97/SC 21, *Estelle - A Formal Description Technique Based on an Extended State Transition Model*, Draft International Standard 9074, 1987.

[2] M. Barbeau, *Estelle: A Formal Description Technique for Communication Protocols*, Proceedings of the 6th ARRL Computer Networking Conference, Redondo Beach, California, August. 1987.

[3] S. Budkowski and P. Dembinski, *An Introduction to Estelle: A Specification Language for Distributed Systems*, Computer Net works and ISDN Systems. Vol. 14, No. I, 1987.

[4] T. L. Fox, *AX.25 Amateur Packet-Radio Link-Layer Protocol*, available from ARRL, Newington CT USA 06111, 1984.

5' J. Harangozo. *An Approach to Describing a Link Level Protocol with a Formal Language*, Proceedings of the Fifth Data Communication Symposium, Snowbird, Utah. 1977.

[6] G. V. Bochmann and R. J. Chung, *A Formalized Specification of HDLC Classes of Procedures*, Proceedings of' the National Telecommunications Conference, Los Angeles, 1977.

'7] G. 'I-. Bochmann, *A General Transition Model for Protocols and Communication Services*, IEEE Transact ion on Communications, Vol. COM-28, No. 4, April 1980.

[8] G. V. Bochmann and T. Joachim, *Development and Structure of an X.25 Implementation*, IEEE Transaction on Software Engineering, Vol. SE-5, pp. 429-439, Spet. 1979.

[9] P. Taylor, *Design Abstraction for Protocol Software*, Proceedings of the 6th ARRL Computer Net working Conference, Redondo Beach, California, August) 1987.

[10] P. D. Amer, F. Ceceli and G. Juanolle, *Formal Specification of ISO Virtual Terminal in Estelle*, Proceedings of IEEE INFOCOM'88, New Orleans, Lousiana, March 1988.

[11] G. V. Bochmann a:nd C. Sunshine, *Formal Methods in Communication Protocol Design*, IEEE Transaction on Communications, Vol. COM-28, No. 4, April 1980.

[12] T. Bolognesi and E. Brinksma, *Introduction to the ISO Specification Language LOTOS*, Computer Networks and ISDN Systems. Vol. 14, No. 1, 1987.

[13] CCITT/SGXI, *Functional Specification and Description Language*, Recommendations 2.101 to 2.104, 1985.

International Code Designator for Amateur Radio

J. Gordon Beattie, Jr., N2DSY

Thomas A. Moulton, W2VY

The Radio Amateur Telecommunications Society

206 North Vivyen Street

Bergenfield, New Jersey 07621

Abstract

Amateur Radio **now** has **an International Code Designator (ICD)** assigned by the International Organization for Standardization **(ISO)** to identify Amateur Radio organizations, network components and applications in Open Systems. Open Systems are defined by their use of the communications protocols of IS0 and the International Consultive Committee for Telephone and Telegraph (CCITT) and their national member bodies. The architecture has been defined by these bodies as a model called **the Open Systems Interconnection Reference Model (OSI-RM)** to facilitate communications between applications using different computers and operating systems.

1. Introduction

The International Code Designator (ICD) for OSI based Amateur Radio Systems, Network Elements and Applications was assigned to the Radio Amateur Telecommunication Society **(RATS)** on behalf of the global Amateur Radio community. RATS is responsible for the establishment of a basic information framework for OSI-based, Amateur Radio systems. This framework will develop during the next few years, but some initial elements are needed for immediate use. The high level portions of the framework have been defined. The initial information framework includes the specification of:

- Domains, including:
  - Nations
  - Individual Amateur Radio stations/operators;
  - National Amateur Radio bodies;
  - Other Amateur Radio bodies.
- OSI Network Service Address Point Identifier (NSAP-ID).

These items are described in the remaining sections of this document.

Future additions will include data elements which are needed in Application Messages, including:

- Interpersonal Mail;
- Bulletins;
- Conferences;
- Files
- Remote Execution.

These items will be described in future documentation.

The next section provides an overview of the ICD. it was obtained. The third section describes the Information Framework. The next section outlines the initial objects in the information tree. These objects are in the branch which defines "domains". The last section describes the Network Service Access Point Identifier which is based on the domains described in the previous section.

2. International Code Designator

The International Code Designator (ICD) for the global Amateur Radio community was issued in April of 1988. The highlights of the request/issue are provided below.

- Name of the Coding System:

  International Code Designator for the Identification of OSI-based, Amateur Radio Organizations, Network Objects and Application Services

- Structure of the code:

  Class Instance

  — The Class is a field which indicates the general category of object such as the identified organization, network element or application service.

  — The Instance is a field which indicates the specific instance of the organization, network element or application service.

- Display requirements:

  The "Class" and "Instance" fields combine to uniquely identify the a particular organization, network element or application service.

- Description of the organizations covered by the coding system:

  Amateur Radio groups which wish to participate in the OSI-based networks.

- The ICD assigned to the global Amateur Radio community:

  0 0 1 1 (Decimal)

3. Information Framework

The Information Framework is a tree-like structure which is designed to distribute the assignment authority for identified organizations, network elements and application services. The peak of the information framework tree is the **"root"** This Information Framework contains objects which are organized into classes and subclasses. These **Object Classes** have **attributes** which may be other classes (subclasses) or parameters. Some attributes of an object are used to unambiguously identify a specific instance within an Object Class. Other attributes provide information found to be useful when referencing an object.

4. Objects in the Amateur Radio OSI Information Tree

The initial branch in the Amateur Radio OSI Information Tree is the domain. The establishment of this object and its attributes will allow for the immediate delegation of assignment authority to groups and individuals within the community. It will also provide a mechanism for the definition of network addresses.

Domains = [0]

An domain is one in which a body such as a country, an amateur radio society or an individual radio amateur defines the components and systems under their authority. The following are defined for the OSI -based Amateur Radio community:

- **National Domains = [0]**

  Amateur Radio is administered on a national basis. The values used to indicate a specific country will be derived from the three digit Data Country Codes in CCITT X. 121 (1988).

- **Individual Amateur Radio Stations/Operators = [1]**

  This subclass includes amateur stations/operators. Assignment of values is implicit because the domain identifier consists of the national regulatory authority (PTT or equivalent) assigned station callsign.

- **National Amateur Radio Bodies = [2]**

  This subclass includes IARU member bodies. Values will be assigned to each body shortly.

- **Other Amateur Radio Bodies = [3]**

  This subclass includes international and local amateur radio societies. The IARU has been assigned the first value in this subclass. Other groups will be assigned values upon request.

5. OSI Network Service Address Point Identifier (NSAP-ID)

The Network Service Access Point ID is used in OSI-based networks to identify the OSI network subdomain and system. The callsign is derived from the Individual Amateur Radio Station/Operator Domain. The Data Country Code is derived from the national domain. Values are encoded as follows:

NSAP-ID Encoding	
Value	Definition
47H	Authority and Format Identifier - ISO, Binary
00H	Initial Domain Identifier (two octets, BCD encoded)
11H	Initial Domain Identifier = Amateur Radio
ll	Length Indicator for Callsign (Binary encoded)
cc	Callsign octets
-	(up to 9 octets, BCD encoded)
cc	
dd	Data Country Code + 0 (two octets, BCD encoded)
dd	
nn	National Number (up to 5 octets, BCD encoded)
nn	

6. Conclusion

The development: of OSI-based Amateur Radio Systems has progressed significantly in the last several years. This is consistent with its growth in the commercial telecommunications industry. Public domain software exists at all seven protocol layers for MS-DOS, Unix and other operating systems. Initial applications include message handling (CCITT X.400), file transfer (FTAM), and directory services (CCITT X.500). These applications and those under development will need an expanded information framework upon which Open Systems can grow. Those interested in participating in this process should contact one of the authors.

Amateur Framing Protocol

J. Gordon Beattie, Jr., N2DSY

The Radio Amateur Telecommunications Society

Terry Fox, WB4JFI

The Amateur Radio Development Corporation

Thomas A. Moulton, W2VY

The Radio Amateur Telecommunications Society

Abstract

Over the last ten years Amateur Radio packet mode operations have evolved to the point where there are approximately fifty thousand Radio Amateurs worldwide using the AX.25 Link Layer Protocol implemented in special computer interfaces called Terminal Node Controllers (TNCs). During this period many functional extensions or changes to the protocol have been suggested, but their use would not conform to the basic AX.25 protocol.

The authors have identified a common set of protocol capabilities which can be provided in a simple framing sublayer. These capabilities include framing, station identification, bit-error detection, digipeating, and application specific options. The Amateur Framing Protocol (AFP) presented in this paper defines a a format and a set of rules to support these functions, while providing an envelope for higher layer protocols such as ARRL AX.25 Link Layer, CCITT Q.921, CCITT 4.93 1, CCITT X.25, IS0 8208, IS0 IP, and US DoD TCP/IP and ARP.

Introduction

The Amateur Framing Protocol is a stable base for the evolving Amateur Radio packet protocols. Its major objective is to provide a format for proper station and protocol identification. It also provides framing, bit-error detection, digipeating, and application specific options.

In this model, the framing sublayer protocol has been defined separately from the logical link layer protocol. The separation of the framing protocol is an outgrowth of the diverse software development work done by the authors and other members of the Amateur Radio community. This framing protocol provides a common base for the development of the ARRL AX.25,[1] the CCITT X.25,[2] and the U.S. Department of Defense TCP/IP[3] and other protocols in the Amateur Radio Network.

The creation of a new framing protocol provides an ideal opportunity to incorporate the various station identification format requirements of regulatory authorities. AFP has also been structured to support local protocol extensions for cooperating systems.

Functional Overview

The Amateur Framing Protocol was developed from a “wish list” of enhancements to the AX.25 Link Layer Protocol which came from the ARRL Digital Committee, vendors, Radio Amateurs worldwide, and the authors. Functional capabilities of the framing sublayer were included in AFP and are described in this document.

- **Hardware Selective Receive**

  In order to reduce the overhead of processing every received frame, AFP requires that the first octet received in a frame match a predefined value which is based on a hash function of the next station’s identification. Only when this octet is matched, does the serial interface interrupt the processor. This leaves the processor free to handle other tasks. This is beneficial when operating on a high-speed channel because many frames can be immediately discarded by the hardware when this octet fails to match. This capability also supports the automatic detection of a “broadcast-mode” frame. This causes the hardware to receive AFP frames with the value 0FFH in the first octet.

- **Version Number**

  This protocol features a version number which will provide a mechanism to indicate major frame format differences to other systems.

- **Upper Layer Protocol Identification**

  This field provides an indication of the upper layer protocol type(s) in use by the system. This protocol identification has assigned values for ARRL AX.25 Link Layer, CCITT Q.921, CCITT X.25, IS0 IP, and US DoD TCP/IP and ARP. Other protocol values can be added after review.

- **Variable Length Station Identifiers**

  AFP supports variable length station identifiers up to a maximum of 40 octets. An additional “length octet” is provided before each Station Identifier field.

- **Simplified Relay (Digipeating)**

  Digipeater stations are primitive source-routed relay devices, this function was incorporated into AFP. A simple pointer is provided to indicate the start of the next receiver’s Station Identifier in the frame header.

- **Supplementary Header**

  This field allows individual protocol implementations to carry addlitional header information such as is required by an administration, or other information required by a specific local protocol implementation.

- **Header Checksum**

  In many applications the overall frame may be received in a damaged condition yet the header might be usable by the system. Minor corruption of the data may be acceptable in applications such as packet voice facsimile or video.

- **Frame Data**

  This field contains the frame of data used on the communications channel. The protocol format for this field is governed by the protocol indicated in the Upper Layer Protocol Identifier field.

- **Frame Check Sequence**

  This field provides a means by which bit-errors can be detected. It contains the sixteen bit Cyclical Redundancy Check as specified by IS0 **3309**[4] **(HDLC).** This is CRC function used in AX.25 and X.25.

Conclusion

The authors have included the text of the Amateur Framing Protocol (AFP) in this paper. This protocol provides a basic tool which can be used by the Amateur Radio community to explore advanced digital communications techniques while maintaining the ability to verify the source of transmissions and providing a basic relay capability. It is hoped that this protocol will act as a catalyst for the rapid development and deployment of a wide variety of communications protocol implementations.

Acknowledgments

The authors wish to thank Phil Kam, **KA9Q,** Paul Rinaldo, **W4RI,** Eric **Scace, K3NA,** and Harlan Worchel, **N2IDS** for their technical input to this document. The authors also wish to thank Nancy **Beattie, N2FWI,** John Siegel, **K4BNC,** and Doug **Zuckerman, W2XD** for their review of the early drafts.

Amateur Framing Protocol Specification

15 August 1988

J. Gordon Beattie, Jr., N2DSY

The Radio Amateur Telecommunications Society

Terry Fox, WB4JFI

The Amateur Radio Development Corporation

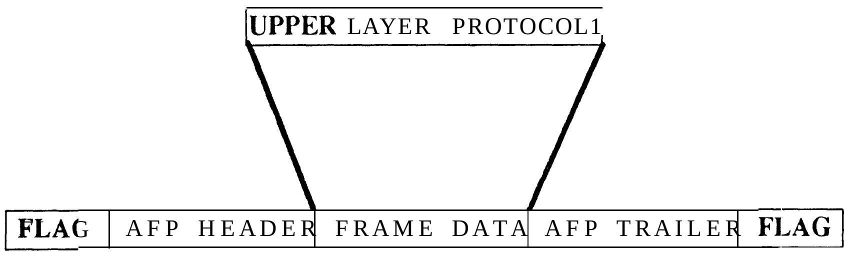
Thomas A. Moulton, W2VY

The Radio Amateur Telecommunications Society

1. Introduction

The Amateur Framing Protocol (AFP) is a general purpose data format protocol for Amateur Radio packet mode communications systems. AFP provides the framework on which other protocols may be implemented.

An AFP frame provides an envelope to support any form of Logical Link Control protocol which two (or more) stations wish to use during packet mode operations. The relationship of AFP to other protocols is shown below.



1.1 Relationship of AFP to Other Protocols

The upper layer protocols currently supported by AFP are listed in section 3.3. These protocols will generally be of the link layer, but may also be network layer protocols or other protocol stacks.

1.2 Document Scope

This document presents the frame format, the field encodings, and other guidelines for users of the protocol.

2. *Frame Structure*

2.1 Frame Format

The basic AFP frame format is shown in the table below.

AFP Frame Format	
Octet	Field
	Flag
0	Next Station ID Checksum
1	Version
2	Upper Layer Protocol ID
3	Frame Data Offset
4	Next Station ID Offset
5 to X	Station IDs
x+1 to y	Supplementary Header
y+1	Header Checksum
y+2 to z	Frame Data
z+1	Frame Check Sequence
z+2	Frame Check Sequence
	Flag

2.2 Frame Boundaries

The Flag Field occurs before and after each AFP frame. Two frames may share one flag, which would denote the end of the first frame and the beginning of the next. The Flag is will contain the value 7EH.

2.3 Frame Length

2.3.1 Maximum

The maximum length of AFP frames is 2560 octets. This value includes an AFP Header length of 254 octets(max.), a Frame Data length of 2304(max.), and a Frame Check Sequence of 2 octets. This does not include the Flag field which bound the frame.

Specific upper layer protocols or local implementations may limit the frame length to a lower number of octets by setting limits on the AFP Header or Frame Data fields.

2.3.2 Minimum

The minimum length of an AFP frame is 14 octets. This includes the Next Callsign Checksum(1), Version(1), Upper Layer Protocol ID(1), Frame Data Offset(1), Next Station Identification Offset(1), Station Identification(6 - if one, four character callsign), Header Checksum(1), and a Frame Check Sequence(2) fields.

2.4 Bit Order

With the exception of the Frame Check Sequence field, all fields of an AFP frame are sent with each octet's least-significant bit first. The Frame Check Sequence is sent most-significant bit first.

25 Octet Order

The octets are transmitted in the order in which their fields occur in the figure in section 2.1.

3. Field Encodings

3.1 Next Station Identification Checksum

The Next Station Identification (Station ID) Checksum octet contains an eight bit value which is the one's complement sum of each octet of the Station Identification field of the next receiver. If the result is 127 (0FFH) for a non-broadcast frame, then the value 0 (00H) should be used. The algorithm used to calculate the Next Station Identification Checksum is:

[SUM(n) = ~SUM(n-1) + OCTET(n)]

3.2 Version

The Version octet presents the AFP Version. The value 1 (0000001B) represents the current value.

AFP Version Octet Encoding		
Version	Binary Value	Hexadecimal Value
1	00000001B	01H

3.3 Upper Layer Protocol ID

The Upper Layer Protocol ID is indicated, by this octet. The values OFEH and 0FFH are reserved for future extensions.

Specific encodings are shown in the table below.

Upper Layer Protocol ID	
Protocol(s)	Hexadecimal Value
CCITT x.25	OOH
IS0 8208	10H
CCITI' Q.921	20H
CCITI' 4.93 1	30H
IS0 IP	40H
DoD IP	OCOH
DoD ARP	0C2H
ARRL AX.25	0F0H
Extension	OFEH
x-ension	OFFH

3.4 Frame Data Offset

The Frame Data Offset octet contains a seven bit index from the beginning of the frame (the Next Station ID Checksum is position 0) to the start of the Frame Data field. The maximum AFP header length is 254 octets. The maximum value is 254.

35 Next Station Identification Offset

The Next Station Identification Offset octet contains a seven bit index from the beginning of the frame (the Next Callsign Checksum is position 0) to the length octet of the Station ID field of the next receiver. The next receiver is either the destination, digipeater or other relay device.

3.4 Station Identification Fields

3.6.1 Amateur Station Identification

Station Identification (Station ID) octets contain the legal Amateur Radio Station Callsign. The legal Amateur Radio Station Callsign must be encoded into the first octets of the field. The Callsign may contain any uppercase alphabetic character (A-Z), and numeral (0-9), or the "slash" (/) character.

Any additional station assigned identification may be appended to the callsign. This additional identification may be used to provide uniqueness of a particular transmitter or to indicate a special capability to other systems. The ASCII character "-" or "Dash" (2DH) is used as a separator between the legal Amateur Radio Callsign and any station assigned identification.

3.6.2 Station Identification Field Order

Station Identifiers occur in sequence in an AFP frame.

The first and only required Station ID is the Source Station ID. Subsequent Station ID fields contain either the Destination Station ID, a digipeater Station ID or the Station ID of another type of relay device.

The last Station ID on a direct or digipeated path is that of the Destination.

If a digipeater (or digipeaters) is in the path between the Source and Destination stations, the Station ID field (or fields) is placed in order of communications in the frame.

The table below presents the layout of the Source, Digipeater and Destination Station Identification fields.

Station Identifier Fields						
Station ID 1		**Station ID 2**		Station ID 3		**End**
Length	Source	Length	**Digipeater**	Length	Destination	End
9	N2DSY-3B 1	9	W2VY-DIGI	8	KA9Q-SUN	*0*

No Station ID sequence restrictions are placed on other types of operations other than that the first Station ID field must always contain the Station ID of the Source station.

3.6.3 Station ID Field Encoding

Each Station ID field contains length octet and a sequence of Station ID octets. The octet following the last Station ID field is encoded with the value 0.

3.7 Supplementary Header

The Supplementary Header field provides a mechanism to include information needed to meet various national, local implementation or Upper Layer Protocol-dependent requirements.

3.7.1 Supplementary Header Field Encoding

The first octet contains the length octet for the remainder of the field. Subsequent octets contain the data of the Supplementary Header field.

The table below shows the encoding of the basic Supplementary Header field.

Supplementary Header Field
Supplementary Header Length Octet
Supplementary Header Data

3.7.2 Supplementary Header Options Encoding

The Supplementary Header contains subfields which contain Supplementary Header Options. Each Option has an encoded in a type, length and value sequence. The values in the range O-127 and 255 are reserved for standard values available to all AFP users. The values 128- 19 1 are reserved for use by specific upper layer protocols. The values 192-254 are available for local protocol implementations.

3.8 Supplementary Header Options

There are two standard Supplementary Header options which are available for all AFP users. These deal with the identification requirements of some telecommunications administrations. The first option provides a means to communicate the legal Amateur Station Callsign of the originating station. The second provides a means to communicate the legal Amateur Station Callsign of the terminating station.

3.8.1 Originating Station Identification Option

This option the callsign of the originating station. The first octet of the field will be encoded with the value 0. The second will contain the binary value of the length of the callsign character string. The characters of the callsign are encoded into a string of ASCII octets.

3.8.2 Terminating Station Identification Option

This option the callsign of the terminating station. The first octet of the field will be encoded with the value 0. The second will contain the binary value of the length of the callsign character string. The characters of the callsign are encoded into a string of ASCII octets.

3.8.3 Supplementary Header Option Encoding

Below is an example of the Supplementary Header Field with both the Originator and Terminator Station ID Options.

Supplementary Header Field with Originator/Terminator Callsigns
Supplementary Header Length
Originating Station ID Option
Originating Length
Originating Callsign
Terminating Station ID Option
Terminating Length
Terminating Callsign

3.9 *Header Checksum*

The Header Checksum octet contains an eight bit value which is the one's complement sum of each octet of the AFP Header octets(Cktets 0 through y, see figure 2.1). The algorithm used to calculate the Header Checksum is:

$$[\ \mathrm{SUM}(n) = {\sim}\mathrm{SUM}(n-1) + \mathrm{OCTET}(n)\ ]$$

If the result is 0 (00H) then the value 127 (0FFH) should be used. The value 0 indicates that the checksum was not calculated by the sender.

3.10 Frame Data

This field contains an octet sequence not to exceed 2304 octets in length, which conforms to the protocol requirements of the protocol indicated by the Upper Layer Protocol ID field

3.11 Frame Check Sequence

The Frame Check Sequence is a 16 bit number, encoded into two octets and calculated by both the sender and receiver of a frame. It is used to insure that the frame was not corrupted by the medium used to get the frame from the sender to the receiver. It is calculated in accordance with IS0 3309 (HDLC).

REFERENCES

1. T. Fox, WB4JFI, "AX.25 Amateur Packet-Radio Link-layer Protocol Version 2.0", The American Radio Relay League, Inc. (1984).
2. CCITT, "CCITT Recommendation X.25", CCITT (1984)
3. J. Postel et al., "DDN Protocol Handbook", USC-ISI (1986).
4. ISO, "ISO 3309", International Organization for Standardization (1981)

A Routing Agent for TCP/IP:
RFC 1058 Implemented for the KA9Q Internet Protocol Package

Albert G. Broscius, N3FCT

University of Pennsylvania
200 S. 33rd Street
Philadelphia, PA 19104-6319

ABSTRACT

The KA9Q Internet Protocol Package has introduced full TCP/IP internetworking to the Amateur packet community yet, until now, automatic routing has not been available within the package. This paper describes an implementation for the KA9Q package of the DDN Internet standard Routing Information Protocol as specified in RFC 1058. Proper usage and configuration of this routing code are explained.

Introduction

Routing in the Amateur Internet has been done so far on an ad-hoc basis with manual table manipulation for routes to both single hosts and to host clusters. The steady-state routing solution will see KA9Q interchanging routing information in a variety of forms such as RIP[1], EGP[2], HELLO[3], IGRP[4], and RSPF[5]. With this array of routing protocols available for use the Ampmet may interoperate effectively with commercial Internet gateway implementations[6].

In this context, RIP is the first of the Internet standard routing protocols to be implemented in the KA9Q package because it may easily interface a packet radio (PR) network to 4.3 BSD Unix-based networks which also use RIP. Guidelines for its use in this application are outlined briefly below in the Configuration section. As the first routing protocol available in KA9Q, RIP will most likely also see use as an inter-PR routing protocol even though its suitability for this purpose is questionable and its use discouraged.

The RIP Protocol

The Routing Information Protocol, as specified in RFC 1058, is a distance vector routing algorithm based on the 4.3 BSD Unix program, "routed"[1]. The basic distance vector routing algorithm consists of a periodic transmission by each IP switch of its internal routing table on the several attached network interfaces. A switch's routing table reflects the costs of sending packets to a set of destinations; each cost is termed a "metric" from the jargon of higher mathematics. The table which is broadcast to a switch's neighbors has its metrics incremented to account for the extra hop that packets would need to take to use the route. Upon reception of a neighbor's transmitted routing table, the metrics of the routes to each of the destinations listed is compared to those now held in the switch's routing table. Those destinations which can be routed through the neighbor more cheaply (lower metric) than the current route are entered into the switch's table and the current route is dropped. After some number of iterations the algorithm converges on a set of routing tables for the participating switches.

RFC 1058 RIP Modifications

The "routed" protocol has been augmented with two policies designed to speed the algorithm's convergence in response to changes in topology. The first of these policies, Split Horizon with Poisoned Reverse (SHPR), attempts to prevent two party routing loops from forming when a destination known to both switches becomes unreachable. SHPR assumes that the subnetworks connected to a switch are themselves fully connected. This implies that any packet arriving at a given network interface should not be forwarded through that interface since this would be a waste of network bandwidth; the proper route would not need to involve the forwarding node but would take the packet directly across the subnet. Since UI frame AX.25 subnets obviously do not qualify as fully connected, this feature of the protocol should be disabled on these subnet interfaces. However, the assumption of full subnetwork connectivity is valid for SLIP, Ethernet, and (ideally) Net/Rom subnetwork interfaces so SHPR should be enabled in these cases. To accomplish SHPR, an IP switch discriminates in its route update messages, telling its neighbors on SHPR subnetworks that all routes currently using this interface are inaccessible through this IP switch. This inaccessibility translates to an infinity metric for all such routes (in this case, infinity is defined to be 16). In this way, IP switch X on a SHPR subnetwork which loses adjacency to a destination will not attempt to route to that destination through neighbor switches whose only route is through switch X.

The second new policy is called Triggered Updates. When an IP switch makes a RIP-induced change to its routing tables, it sends a route update with only the affected routes very soon afterward to all of its neighbors. This update is unconditional and must be performed regardless of the status of the pending periodic route update. A small, random delay is inserted before the triggered update is performed to prevent a small network pileup. Effectively, these triggered updates force rapid convergence of the various IP switches' routing tables as the cascade of updates only traverses the tree of nodes dependent on the original affected route. This protocol feature may also be disabled on a per-interface basis if desired, Note that it is recommended that this feature be enabled for all subnetworks known at this time.

Optional RIP Additions Implemented for the KA9Q Package

Self Announcement

In the land-line Internet the main use of RIP is the announcement of subnetwork reachability by gateways attached directly to those subnetworks. Each subnetwork consists uniformly of hosts and gateways having a subnetwork IP address with a fixed prefix of a given length. Gateways belonging to several subnetworks are given an IP address for each of their attached interfaces. In this system, routes to individual hosts are the exception and subnetwork routes predominate in the routing tables.

Where the wired intemet has a clearly defined topology, the Amateur PR internet (amprnet) has few limitations on, and correspondingly few expectations of, the realized network topology.. After much discussion [7], it was decided that the amprnet IP address assignment should not reflect any correlation between IP address and subnetwork medium/frequency/modulation technique/etc. The IP addresses should reflect only the region-of-issuance of the station address as assigned by the regional IP address coordinator [8]. This flat IP address space approach has been used successfully in the NSFNET Backbone implemented with the Distributed Computer Network (DCN) architecture [9].

Since the subnetwork grouping of the land-line Internet may not be used in the amprnet, the RIP processes have the option of advertising a route to their own IP address. This option is configured on a per-interface basis. It is suggested that any Amateur constructing a personal, wired IP subnetwork obtain a Class C network address from the DDN management at SRI-NIC[lO] to lessen the routing space requirements of the amprnet. Since all of the hosts on the wired subnet will be routed as a single entity, less network overhead is needed for this group as a Class C network.

Private Routes

Provision has also been made for the construction of private routes which are used as normal by the IP router module but are not advertised to neighbors in route update messages. These routes could be used for intentional partitions of the network where organizational boundaries necessitate it. For example, a university-to-PR gateway may know the route to net 10, the ARPAnet, but it may not wish to tell other PR hosts that this route exists. It then uses a private route to facilitate its own communication without disclosing to others the route's existence.

Neighbor Refusal

Occasionally, there arises undesirable one-way or unpredictable propagation paths which introduce routes of dubious value into the network. These routes may be suppressed with the judicious use of neighbor refusals. A neighbor refusal list is maintained which is compared against incoming RIP datagrams. If a match is found the datagram and its contents are discarded.

KA9Q Package RIP Configuration

For each network interface which is to be used to transmit route update messages an ARP entry must exist in the appropriate table. In the case of subnetwork broadcasts, this entry must be installed manually in the autoexec.net file to associate the hardware broadcast address with the IP subnetwork broadcast address. For example, to participate in routing on the local Ethernet here at UPenn I must add:

```
arp add 128.91.0.0 ether ff:ff:ff:ff:ff:ff
```

to enable subnet broadcasts onto the Ethernet and also

```
route addprivate 128.91/16 ec0
```

to have my IP switch route local network traffic onto the Ethernet.

To start the routing agent itself, the following command is also added:

```
rip init 128.91 .O.O
```

which opens the RIP UDP socket locally to listen for route updates. It also broadcasts a request for the default router on this network. If no default router is available/desired, this command may be abbreviated to

```
rip init
```

which will send no broadcast request. At this point the routing agent is still silent, only listening for route updates from the routers conversing on the local network. To enable the routing agent to participate in the conversation, the "rip add" command is used. The general form of this command is:

```
rip add <destination-address> <iface> <interval> <flags>
```

where <destination-address> is the broadcast or host address of the neighbor which should receive route updates, <iface> is the interface over which we will receive route updates from this neighbor, <interval> is the amount of time in seonds we should wait between route update messages to this peer, and <flags> is a set of three bits which enable/disable SHPR, Triggered Updates, and Self Advertisement (from msb to Isb). The configuration here at UPenn has a rip add of the following form:

```
rip add 128.91.0.0 ec0 30 6
```

which causes the rip agent to output route update messages every thirty seconds (as per RFC 1058) to the local network broadcast address and to assume that route update messages received by interface ec0 have been generated by a thirty second interval rip agent also. The flags, decimal 6 = binary 110, designate that this interface Will perform SHPR, Will perform Triggered Updates, and Will Not perform Self Advertisement.

To ignore route update messages from certain other routers, the "rip addrefuse" command is used. If there were a misbehaving local router which I wished to suppress, say for example 128.91.254.34, I would place a

```
rip addrefuse 128.91.254.34
```

in my autoexec.net to force the routing agent to drop all route update messages from 128.91.254.34 which may arrive.

References

1. Hedrick, C., "Routing Information Protocol - RFC 1058'', SRI Network Information Center, 1988.
2. Mills, D., "Exterior Gateway Protocol Formal Specification - RFC 904'', SRI Network Information Center, 1984.
3. Mills, D., "Distributed Computer Network Protocols - RFC 891'', SRI Network Information Center, 1983.
4. Reynolds, J. ed., "Assigned Numbers - RFC 1010'', SRI Network Information Center, 1988.
5. Goldstein, F., "Radio Shortest Path First (RSPF) Routing Protocol Specification"', as yet unpublished, 1988
6. Postel, J. ed., "Requirements for Internet Gateways - RFC 1021'', SRI Network Information Center, 1987.
7. Tcp-Group electronic mail conference, Discussion on meaning of IP address assignment in network topology, 1987.
8. Each metropolitan area or foreign country with IP activity has been assigned an Ampmet address block. Station IP addresses are to be assigned by a coordinator in each region. The list of coordinators is available from:

   Brian Kantor, wb6cyt
   University of California, San Diego
   Academic Network Operations Group
   Mail Code B-028
   La Jolla, CA 92093

   brian@ucsd.edu

9. Mills, D. & Van Braun, W. "The NSFNET Backbone Network", Proc. ACM SIGCOMM, Stoweflake, VT, August 1987.
10. Application for an independent Internet network number can be made to:

    Hostmaster
    DDN Network Information Center
    SRI International
    333 Ravenswood Ave
    Menlo Park, CA 94025
    l-(800)-235-3 155

    HOSTMASTER@SRI-NIC.ARPA

RECOMMENDED POWER AND ANTENNA HEIGHT GUIDELINES FOR LAN's AND WAN's

Bob Bruninga WB4APR
59 Southgate Avenue
Annapolis, MD 2,401

What follows is simply an appeal that we apply some degree of frequency coordination within the digital allocations on two meter FM. We have noted rapid growth in the WASH DC area as shown in figure 1 with over 38 BBS's, 35 DIGI's and a number of NET/ROMS and TCP/IP nodes spread over the 100 KHz segments starting at 145.0, 145.5 and 145.6 plus 221 MHZ. The nature of packet radio is quite forgiving in accommodating multiple users and a mix of services on any one frequency. But condoning a total free-for-all mixture can not possibly result in an effecient network. The opposite extreme of total coordination and rule making is restrictive and abhored by most radio amateurs.

Figures 2 and 3 show the two extremes of purely LOCAL LANs and WIDE area LANs. What I hope to show is that local LANs should be kept relatively limited in range and that wide area LANS may cover as much area as they need. BUT THAT THE TWO FUNCTIONS SHOULD BE ON SEPARATE FREQUENCIES to optimize the effeciency of both.

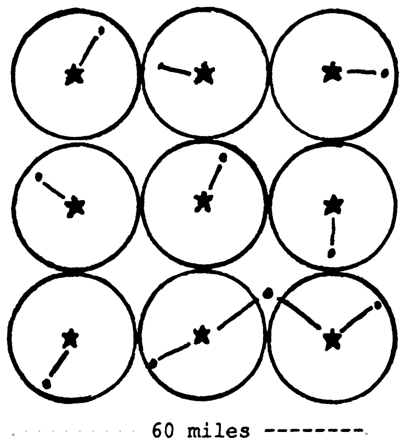


Figure 1. Nine local area LANs on the same frequency with 9 users use-ing 9 different local BBSs with no contention. There is no hidden transmitter problem and each user gets 100% of the channel during his session.

MAKING LAN'S AND WAN'S WORK

To make cellular work, however, there has to be cell-to-cell or LAN-to- LAN connectivity. This is where the wide area LAN of figure 2 plays its best role. The wide area full dux repeater is an optimum solution to moving LAN-to-LAN traffic in this example. The wide area repeater is also optimum where a given community of users whos traffic statistics look like a LAN are widely geographically distributed. There is an equal need for this capability in the area.

Figures 2 and 3 show the same users and same service areas but to have the same performance, the Full-Dux repeater and all users and services of figure 3 will have to operate at 8 times the data rate as those same users in figure 2. Since the Ful-Dux repeater takes two freqs, the overall effeciency in this worst case example is 16 to 1 in favor of the figure 2 approach. Ma Bell was no dummy when she invented cellular!

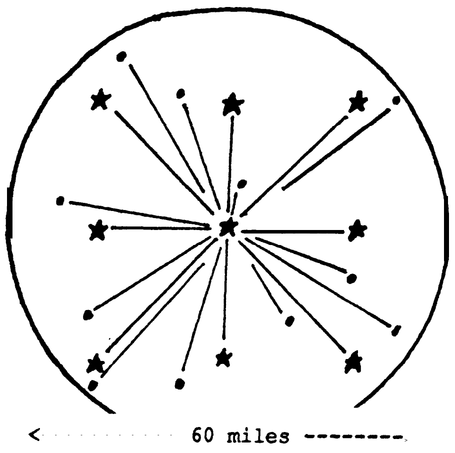


Figure 2. One wide area WAN with eight users using 8 different BBSs through a wide area digipeater or node. A full dux repeater is required to solve the hidden transmitter problem.

There should be no argument that delivering mail via the present BBS systems is a purely local function. Our goal should be that every HAM everywhere has access to at least one mail drop system. We have reached that condition in the DC area and we should optimize that function, but not at the expense of others, through reasonable frequency management. With most BBS software supporting multiple ports and frequencies, the separation of user access from forwarding channels is going well and must continue to be encouraged. This establishes the basic cellular LAN structure.

RECOMMENDED LAN GUIDELINES

The second part of the cellular equation is minimizing interference through geographic distribution and power limitations. Since most BBS stations are located at home stations with typical antenna heights, they serve as a good cell center model. They should be balanced with their users so there is no need to have a lkw power level if the user is typically only running 25 watts or less. The following recommended power levels for BBS's digipeaters, NET/ROMS, TCP-IP and any other 24 hour service on the LAN frequencies provide a service area of over 450 square miles (12 mile radius). This coverage limitation should not be too restrictive to the typical ham station which is being used as a LAN service provider, but it does discourage the installation of aligators on LAN frequencies which are disruptive over large areas. A 100 watt digipeater with 6 db gain antenna at 1600 feet QRM's an area of over 20,000 square miles!

ANTENNA HEIGHT ABOVE AVERAGE TERRAIN	RECOMMENDED POWER LEVEL
1000	0.1 w
300	1 W
100	10 w
75	25 W
50	40 w
25	150 w

Similar to conventional voice repeater frequency coordination, this proposal places no restrictions on who can operate on LAN frequencies, only on the area of their influence. In fact, all forms of digital services are equally welcome including BBS's, digis, NETROMS, repeaters, and TCP-IP; although some of these would be less useful than others under the LAN restrictions. Home users could even be encouraged to leave their stations in the unattended digi mode to assist connectivity within the LAN. Anyone may also operate his station at any power level and at any height but not as an unattended service provider on the LAN frequency. Direct BBS forwarding would be permitted where needed on LAN frequencies, but longhaul, digipeated and bulletin forwarding should be limited to non-prime hours or moved over to the WAN frequencies.

This proposal has no intent to establish particular cells or to provide exclusive protection for any particular LAN, but to simply provide a protective framework for the LAN concept. A few frequencies are required so that multiple services in a close geographic area can choose separate frequencies. It would seem that 4 or 5 frequencies should fill this need.

RECOMMENDED PACKET 2 METER BAND PLAN

144.91	reserved for voice (no packet)
144.93	reserved for voice (no packet)
144.95	reserved for voice (no packet)
144.97	reserved for voice (no packet)
144.99	reserved for voice (no packet)
145.01	WAN longhaul primary
145.03	WAN regional
145.05	WAN regional
145.07	WAN regional
145.09	WAN longhaul
145.51	LAN guidelines
145.53	LAN guidelines
145.55	LAN guidelines
145.57	LAN guidelines
145.59	LAN guidelines
145.61	WAN longhaul
145.63	WAN longhaul (possible FDUX out)
145.65	WAN regional
145.67	WAN regional (possible FDUX out)
145.69	WAN longhaul

WAN GUIDELINES

Finally, this proposal cannot work without wide area systems. Without WAN's, packet radio would appear too restrictive to the average HAM who wants to exploit his full potential. All WAN initiatives should be strongly supported and encouraged. There should be no restrictions on the power and range coverage of WAN's although careful planning and coordination can be particularly productive in improving the performance of separate WANs used for longhaul, backbone, mail forwarding, and wide area access. Separating WANs and LANs will help the HAM packeteer have the best of both worlds.

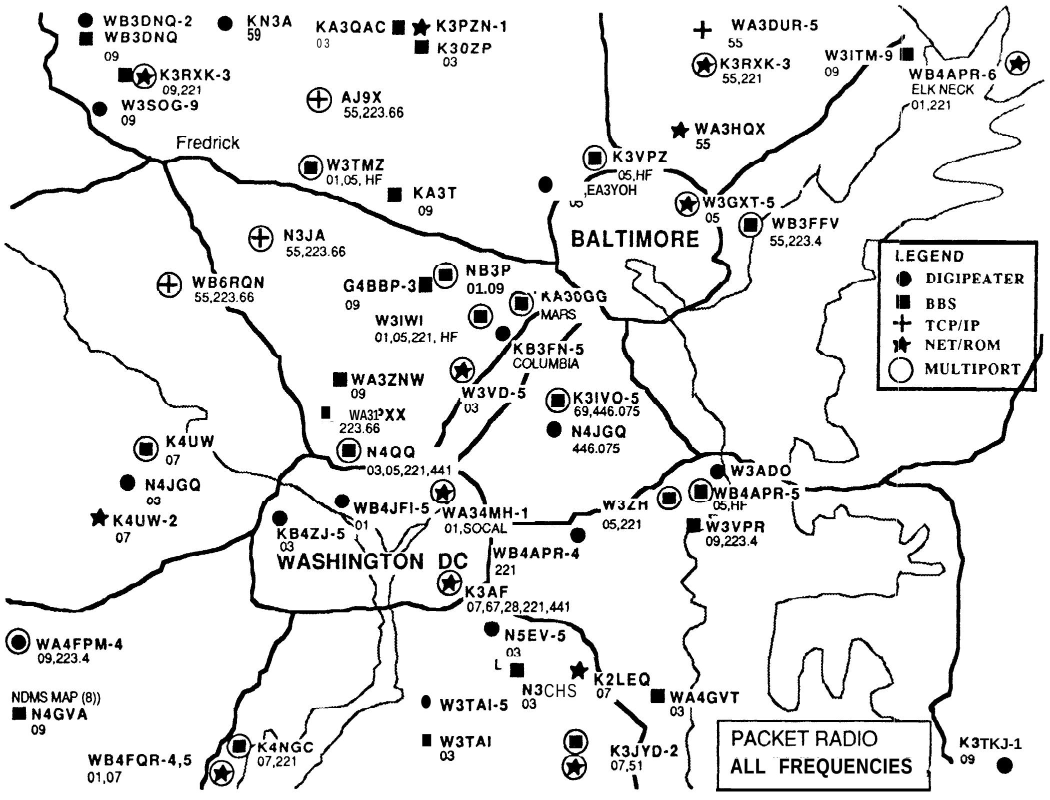


Figure 1. Permanent packet services in the Washington DC and Baltimore areas. Eight two meter frequencies three 220, and three UHF frequencies are in use. So far there are no agreed upon frequencies for LANs only.

A Totally Awesome High-Speed Packet Radio I/O Interface for the IBM PC XT/AT/386 and Macintosh II Computers

Mike Chepponis, K3MC

Bernie Mans, AA4CG

ABSTRACT

This paper describes a plug-in card for IBM PC XT/AT/386 compatible computers or the Apple Macintosh II computer. It is designed to handle two 56 kilobit/sec full-duplex channels via DMA and 10 slow speed (19,200 bits/sec or less) channels via interrupts without main processor intervention. The board uses an NEC V40 processor and up to six Zilog 85C30 Serial Communication Controllers, together with either 256k bytes of DRAM or 1 megabyte of DRAM to offload the main processor from low-level interrupt fielding. It communicates with the main processor with an 8k byte memory window in the IBM version, and directly with Macintosh II main memory using the Bus Master concept of the NuBus. It is targeted at replacing existing TNCs in the high-speed networks of the future; special consideration has been given to the support of TCP/IP. Even the minimum (easily upgradable) implementation with only one 85C30 outperforms all existing TNCs, relying on the host PC only for bootstrapping, power, and of course, an effective user interface when called for.

History

The history of packet radio is replete with ideas, some great, some not so great. One thing we did with this project was to survey the existing solutions for serial communications with a host computer, and to generate a solution that did not have any of the handicaps of previous methods.

In the beginning, folks used TNCs, usually running code that expected the user to attach a "dumb" terminal to them on one end, and to attach an AFSK 1200 baud Bell 202-compatible modem to an ordinary FM radio via the external speaker and microphone jacks.

When Phil Karn, KA9Q, wrote TCP/IP software for the IBM PC XT/AT (and it was subsequently ported to many different machines, including the Macintosh, Atari, UNIX[1], and Amiga), he initially used "nailed-up" AX.25 connections, that is, ordfnary TNCs in the Transparent mode. The disadvantages of such an approach was obvious to all of us, and a simplified TNC interface designed specifically for communicating with computers (the so-called "KISS" TNC[2]) was implemented. The KISS interface allowed the host computer to completely control the TNC, without the bothersome command interface optimized for human use, and it quickly became the "way to go" to use TCP/IP on the air. Still, it used 1200 baud Bell 202-compatible modems on the radio side. Indeed, the KISS TNC was never intended to be anything but a stopgap measure.

Last year, Dale Heatherington, WA4DSY, designed a reproducible 56 kilobit/sec modem[3]. With the help of Doug Drye, KD4NC and a

1 UNIX is a trademark of AT&T Bell Laboratories.

2 Chepponis, M., and Karn, P., "The KISS TNC: A simple Host-to-TNC communications protocol," *ARRL Amateur Radio Sixth Computer Networking Conference,* pp. 38-43, Redondo Beach, 29 August 1987.

3 Heatherington, D. A., "A 56 Kilobaud RF Modem," *ARRL Amateur Radio Sixth Computer Networking Conference*, pp. 68-75, Redondo Beach, 29 August 1987.

host of other Georgia Radio Amateur Packet Enthusiasts Society (GRAPES) members, modem board sets, complete schematics, board layouts and parts lists became available at very reasonable cost. Since then, GRAPES has also made a complete 56k kit available. With the introduction of this modem, Dale needed to figure out a way to connect it with some packet radio hardware and software. Since the TNC-1 could not handle 56k bits/sec, and because the existing TNC code for the TNC-2 was unavailable, Dale chose to perform substantial modifications to the KISS TNC-2 code to permit it to handle 56k data. Also, since no other networking software could handle the requirements of this data rate, Dale chose Karn's TCP/IP software, because it already worked with the KISS TNC, and needed no modifications to run at this higher speed. Because the TNC was essentially a synchronous to asynchronous converter, the 56k radio side was converted into a 19.2 kbaud serial data stream and the effective throughput was limited to only 19.2 kbaud. This is how all 56k modems (to our knowledge) still operate.

What we wanted was true 56k baud system throughput, and this mandated an auxiliary I/O processor if we wanted to use the popular IBM PC XT/AT/386 and Macintosh II hardware. To be sure, four other cards plug into the IBM XT bus, eliminating the need for a TNC: the HAPN card, the 8530-based Eagle card, the Pat-Comm PC-100 and the DRSI card. All of these cards are not appropriate for these higher data rates because none of them have DMA nor on-board CPUs.

There are two other products that should be mentioned: the TAPR NNC and the PS-186. The TAPR NNC did not take off because it was underpowered for the jobs we had intended it to do. The PS-186, on the other hand, is an excellent choice for mountain-top, solar-powered IP switches, and other uses that require low power consumption and several medium-speed channels.

Our solution is particularly cost-effective. Utilizing the very inexpensive IBM PC/XT compatible systems available today with this I/O card costing approximately an additional $200, we can provide true 56 kilobit performance at a very reasonable cost. If one compares the cost of a comparable 1200 baud station, in terms of dollars per bit per second, the 56k solution is indeed *very* cheap!

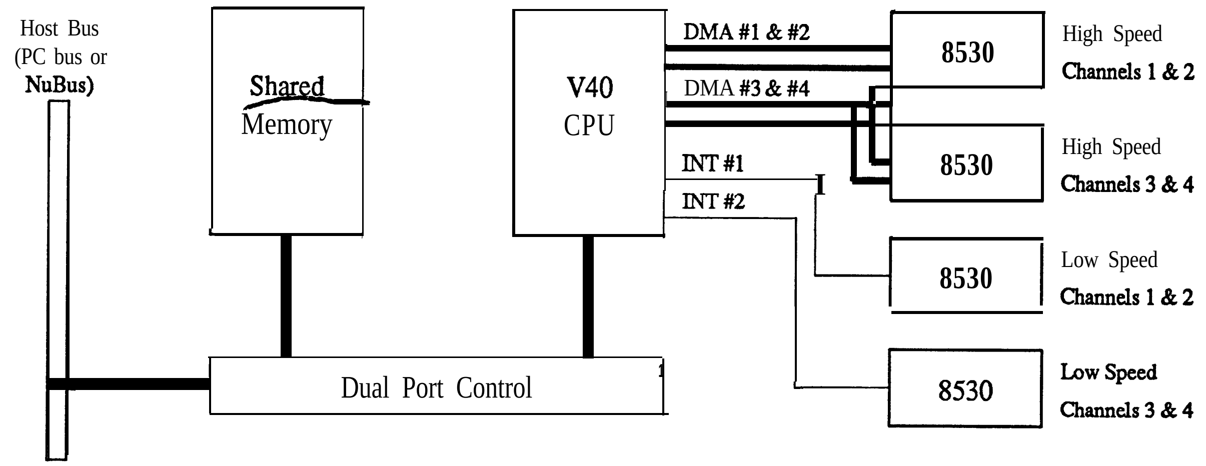
Justification

The board consists of an NEC V40 microprocessor, at least one 85C30, and at least 256k DRAM, all running at 8 MHz. The V40 provides, most importantly, instruction set compatibility with the existing IBM PC XT/AT environment (indeed, the V40 instruction set is a superset of the 8088 instruction set, and contains many instructions of the 80286 processor). It also provides four DMA channels (these four channels are used to provide DMA for the 85C30, allowing for very low processor overhead when all four channels, 2 input and 2 output, are running 56 kilobits, full duplex). There are three 16-bit counter/timers, 8 prioritized interrupts, and an NMI input. In addition, it provides DRAM refresh support. Running at 8 MHz, this chip provides ample horsepower to handle five more 85C30s, for ten more full-duplex channels, using an interrupt-driven scheme, for baud rates of 19,200 or less. An additional pair of 56k channels, full duplex, can be handled, for a total of' four full-duplex 56k channels, if we forsake the low-speed channels; in this case, two of the full-duplex channels would be the standard DMA-driven ones, and the other two full-duplex channels would be interrupt-driven.

The 85C30 is a CMOS version of the ultra-popular Zilog 8530 Serial Communications Controller chip, which is well known in the Amateur community, is highly flexible and very popular. It is used in the FAD PAD, the Eagle card, the DRSI card, the Pat-Comm TNC-220 and AEA's PK-232. With on-board digital PLL clock recovery, on-board baud rate generation, and multiple serial communications formats, it is indeed the "Swiss Army Knife" of serial communications controllers. In addition, the 85C30 can provide a single half-duplex T1 channel (1.544 megabits/se& when we have modems that run at these rates.

256k bytes of DRAM permit about 30 seconds of data buffer for a single 56k channel. When all four 56k channels are active (two input, two output), this memory provides more than 15 seconds of buffer per channel. This permits the host to be relatively lax about servicing this I/O card, and still not miss packets. It permits full utilization of the bandwidth available with the WA4DSY 56k modems.

If the (up to) ten extra low-speed channels are desired, provisions are made on the board



Simplified block diagram for the "Awesome" I/O Interface.

for installation of AMD 7910 modems, or TI 3 105 modem chips.

How it works

Basically, the card is a separate I/O processor, a computer on a board. In the case of the IBM PC XT/AT/386, an 8k byte memory window is used to communicate with the host. Several I/O ports are used for configuring the device. For example, the 8k byte memory window can be made to appear, under software control, into any available 8k byte window in the IBM PC's address space. This is done by using an I/O port with the high-order bit being a "memory enable" bit, and the remaining 7 bits used as the upper 7 bits of the address of where one wants the memory to appear.

The interface is extremely flexible. Another I/O port has a single bit which is tied directly to the V40 reset pin. Upon power-up, the 8k byte memory window is disabled, and the V40 is kept reset. The host enables the memory and places it into the address space where it is desired. Then it loads code into the 8k byte window, and then releases the reset pin by writing into the other host I/O port. At this point, the V40 is running.

With 256k bytes memory, the upper two address bits out of the V40 are ignored. This has the effect of mapping the 256k bytes into the V40's address space four times. The advantage of this is that when the V40 reset wire is held high, it jumps to location FFFFO. Since the 8k byte memory window into the host is always mapped into V40 addresses FE000 to FFFFF, it is trivial to initialize the I/O processor. Also, since the interrupt vectors for the V40 begin at location 00000, all of the interesting parts of the address space are made available without special tricks.

The 8k byte window is dual-ported with the V40 and host processor. In addition, the host always has priority when accessing this memory. It is occasionally necessary to insert wait states into the host processor's access requests, but since the V40 is mostly DMA driven by the 85C30, the host waits on the average only two T cycles, or 250ns at 8 MHz. This means, for example, that a lk byte packet can be transferred from the I/O card to the host in only 770 microseconds, on average.

In the Macintosh II the situation is even better. The NuBus, the bus used on that machine, is capable of having Bus Masters. A Bus Master can seize control of the bus, and perform data transfer operations between itself and either main memory and/or other I/O devices. One way to think of it is as a superset of DMA capabilities; perhaps one could call it "smart DMA". The Macintosh II writes the addresses into the board where to retrieve data (for transmitting) and where to deposit data (for receiving) and this board takes care of the rest! This means that the Macintosh II computer can be doing many more things, because it does not have to move memory blocks around like it needs to do in the IBM PC case. It also means that the 256k byte on-board memory buffer is not required to be quite so big, as the Macintosh II main memory can be used as the buffer.

One may ask why we didn't use DMA on the IBM PC XT/AT machines. In addition to the limited number of DMA channels, the variable latency time of DMA servicing on the IBM machines complicates things, such that we would need a very sophisticated buffering scheme if we were to be certain that bytes were not dropped due to other I/O (especially hard disk or floppy activity) on the IBM PC's bus.

How to use it

The I/O card is particularly easy to program. We have identified two classes of programming, the lowest-level driver code and the application code, which makes use of the low-level drivers.

For the low-level code, we have kept things as general as possible (practicing the Computer Science principle of "delayed bindings") - that is, we have fixed very little about how one should program the card. We have fixed two host I/O addresses, as mentioned above, one for enabling the 8k byte memory and placing where desired in the host address space, and one for controlling the V40 reset wire. Other than that, the low-level programmer is free to define how to use the 8k byte memory window. In particular, which memory locations are used to specify which 85C30 channels, which memory locations are used to hold pointers and length-of-packet values, etc., are all flexible. Indeed, the entire structure is flexible from the low-level programmer's point of view. We do interrupt the main processor when the I/O card needs attention (and this interrupt is strappable so you can use a free interrupt of your host), such as when we've received an incoming packet or when we've finished transmitting a packet. But, given the amount of buffering we've provided, the host need not respond instantaneously to this interrupt. In addition, how the interrupt's reason is communicated to the host via the memory window is completely left up to the low-level programmer, enhancing flexibility.

For the applications programmer, three packages are available for the I/O Toolkit: one for initializing the V40, one for receiving a packet and one for transmitting a packet. These packages interface with both Aztec C and Turbo C for the IBM PC, and with Lightspeed C and MPW C for the Macintosh II. Of course, complete source code is provided with the Toolkit routines".

Applications

We have concentrated on the TCP/IP use in "standard" packet radio, but here are some other things we are planning to do with these cards.

The first thing is to build a complete 56k network node, based on Phil Karn's proposal[5]. Due to the stinging loss of 220 to 222 MHz, our options are more limited, but nevertheless, we intend to build a three-port IP switch, with channels on .43, .902 and 1.2 GHz. Two of these frequencies would be receive-only frequencies, and the other would be a transmit-only frequency, for one of the switches. The other switches in the set would have complimentary transmit / receive frequencies. Such a scheme guarantees that no collisions would take place, and with sufficient link margins, would assure perfect transmission/reception at all nodes.

Other uses are also apparent. For instance, full color digital SSTV with 256 x 256 resolution and 6 bits each of red, green, blue takes less than 30 seconds to transmit. Error free reception is possible, given that the picture is digital, transmitted with error detection and retries. In TCP/IP, the File Transfer Protocol could be used for pictures that must arrive, error-free, or UDP, with less-overhead, could be used if some loss or errors could be tolerated.

Digital voice is yet another interesting possibility. With the Delanco-Spry DSP board or the new 320C25-based DSP board that TAPR/AMSAT is working on plugged into a PC with the I/O processor card, audio from a microphone and preamp could be digitized by the DSP board, compressed, shipped via UDP on the RF link, then uncompressed and converted back to audio. "Voice mail" would be a real possibility with such systems!

4 Indeed, if there is one distinguishing feature common with all TCP/IP experimenters, it is the perceived duty to provide complete source code for *all* of our programs, free, as we believe that others can learn from our code and can further enhance it and re-release it back to the Amateur Community, in the best spirit of Amateur Radio.

5 Karn, P., "A High Performance, Collision-Free Packet Radio Network," *ARRL Amateur Radio Sixth Computer Networking Conference*, pp. M-89, Redondo Beach, 29 August 1987.

The DSP board could also act as modem for the I/O card. It would be easy to experiment with different modems, all digitally realized within the DSP card, choosing the best one for the transmission medium.

And we've only begun to explore the many applications for this high-speed interface.. . What we hope what we have communicated is that *real* 56k bits/sec allows much, much, more than simply faster bulletin board access or quicker "hunt-and-peck" ham-to-ham packet communication. A brave new world of distributed networks and file systems is open to the amateur with this interface and high-speed RF data links!

Status Updates

Those interested in the an up-to-date status report on this project are welcome send electronic mail via usenet or the Internet to mac@leg.ai.mit.edu or !pitt!aa4cg!bernie. You may also dial-up host AA4CG directly at 904/795-3211 at 2400, 1200, or 300 baud, eight-bits no parity. Login as user "packet" and password "radio".

Acknowledgements

We would like to thank Phil Karn, KA9Q, for suggesting the major features of this I/O processor, especially his comment "Design it like an Ethernet card[6]" We also thank Bob Hoffman, N3CVL, for his expert typesetting, once again.

6 Indeed, we used the Western Digital WD8003E as our model.

AMSAT's MICROSAT/PACSAT PROGRAM

Tom Clark, W3IWI
6388 Guilford Rd.
Clarksvilie, MD 21029

ABSTRACT

In 1989 AMSAT-NA plans to launch the first of a series of low-earth orbit (LEO) satellites dedicated to serving digital store-and-forward message handling. These satellites are quite small cubes,, approximately 230 cm (9 inches) on a side weighing less than 10 kg; this small size has led to our calling the project MICROSAT. Despite the small size, the satellites are crammed with state-of -the-art electronics. This paper will review the development program leading to this design and some of the technical details as well as describing how the terrestrial user will make use of the resource. We are planning on the launch of 4 satellites using MICROSAT technology into LEO in early 1989, and several more launches over the next 2 years.

A BIT OF HISTORY

In October 1981, the ARRL, AMRAD and AMSAT jointly hosted the first Networking Conference when packet radio was in its earliest period of development. Doug Lockhart (VE7APU) and the VADCG group had put the first TNCs into our hands. Hank Magnuski (KA6M) and the PPRS had the first digipeater on the air. In the D.C. area a few of us (W4MIB, WB4JFI, K8MMO, W3IWI, KE3Z) were on the air making funny sounds. The seed was planted!

On a warm sunny afternoon the following spring, at the AMSAT lab at NASA Goddard, I took Jan King (W3GEY) aside and told him of an idea I had. At the time we were building the AO-1Ø satellite which was to provide global scale communications from its vantage point in high earth orbit (HEO). My idea was to provide similar communications coverage from LEO using digital store-and-forward techniques, albeit not in real time. The basic idea was for the sender to uplink a message to the LEO satellite; then at a later time when it was in view of the recipient, it would be forwarded to him automatically.

After some more design work, I enlisted the aid of Den Connors (KD2S) who was then spearheading the effort in Tucson which became known as TAPR. Den and I started beating the bushes for support for the program. When the ideas became known to AMSAT, some of the old timers accused us of having lost our minds with statements like “There aren’t more than a couple of hundred people on packet. Packet radio will never amount to anything. etcetera etcetera”. By the fall of 1982 we were starting to see some ground-swell of support, so Den and I scheduled a special meeting (to be held in conjunction with AMSAT’s annual meeting) which was to get inputs from packeteers in several groups on the PACSAT concept. The second purpose was to try to see if we couldn’t come up with a national protocol standard; the result was the adoption of AX.25 (for which some people STILL blame me!).

Soon thereafter we found a potential sponsor who needed PACSAT support to aid in disseminating information on technologies appropriate to developing countries and thus was formed a tie between AMSAT and the Volunteers in Technical Assistance (VITA) and Gary Garriot (WA9FMQ). The VITA PACSAT project enlisted the assistance of Harold Price (NK6K), Larry Kayser (VE3QB/WA3ZIA, now VE3PAZ) and a number of others. The VITA/PACSAT team decided to test their messaging concepts on a UoSAT spacecraft resulting in uo-1 l’s Digital Communications Experiment (DCE). The partnership between VITA and the UoSAT group has continued, and the UoSAT-D spacecraft (to be flown at the same time as our Microsats) is the culmination of that effort.

In the meantime I told the Miki Nakayama (JR1SWB) and Harry Yoneda (JA1ANG) of JAMSAT of our design concepts. The JAMSAT/JARL team were able to implement many of these ideas in the mode "JD" hardware on the Japanese JAS-1 (JO- 12) satellite. They also developed state-of-the-art reproducable 1200 BPS PSK demodulator designs which have become important for future spacecraft designs. Unfortunately the negative power budget on JO-12 has limited the utility, of an otherwise excellent spacecraft.

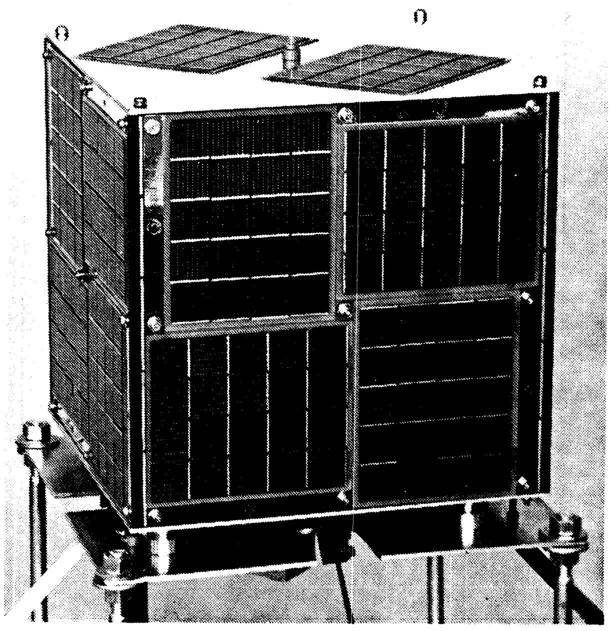


Figure 1. A photograph of the structural mode of the MICROSAT satellite.

For the next couple of years any idea of our building a PACSAT in the USA languished. First we were busy building the AO-13 satellite in consor with AMSAT-DL The American dependence on the Space Shuttle and the lack of suitable launches on which we could hitchhike made opportunities few and far between. We looked at low-thrust motors using water or Freon propellants to lift us to a suitable LEO if we used the Shuttle's GASCANs. Two groups flew small satellites ejected from GASCANs on the shuttle; one was NUSAT, built by a of students and faculty at Weber State College in Ogden, Utah. Then with the loss of the Challenger, even those hopes for our building a PACSAT were dashed.

THE BIRTH OF MICROSAT

The scene now shifts to November, 1987 in a hotel room in Detroit after the banquet at AMSAT's annual meeting. Jan King, Bob McGwier (N4HY), Phil Karn (KA9Q) and I are sitting around at 1AM. Jan starts telling us of a concept that he and Gordon Hardman (KE3D) have been thinking about. It involves a very small, simple satellite, a 9" cube. He describes how five 8" x 8" x 1.6" module "trays" would be stacked to make up the inner frame of a satellite. Then on the small 9" x 9" solar panels would make up the outside skin. He told us that he believed he had several different potential launches that could carry several of these cubes to LEO and asked us what we could do with the limited space. By 3 AM we had a conceptual design, we had done link margin calculations, we had selected a candidate CPU, and we had estimated size, weight and power requirements for each of the modules. The adrenalin flowing in our veins was at an all-time high!

By early December we had refined the basic design. Dick Jansson (WD4FAB) had done a complete mechanical design. We held a preliminary design review at the AMSAT office and decided we were GO!

While all this was going on, contacts were made with Junior DeCastro (PY2BJO) of the Brazilian BRAMSAT group, Arturo Carou (LU1 AHC) of AMSAT-LU and with the NUSAT group at Weber State. Each agreed to join the team and we settled on building four satellites: The AMSAT-NA and AMSAT-LU satellites would be classical PACSATs. The Weber State satellite would be a PACSAT augmented by a TV camera which would send down pictures encoded in normal AX.25 packet frames. The Brazilian satellite would be the DOVE (Digital Orbiting Voice Experiment) which would "talk" voice bulletins which could be copied on a normal HT.

PACSAT AND ALOHA

First we need to review a little packet radio theory. Let us assume that the satellite operates with its transmitter and receiver on different bands so that the communications links are full-duplex. Let us also assume that there are many users, each with similar capabilities, who are spread out over the entire spacecraft "footprint". Let us further assume that traffic is balanced -- whatever goes up to the spacecraft equals what comes down, so the uplink and downlink channel capacity needs to be balanced.

Since the ground-based users are spread out, the cannot hear each other. Each will transmit at random in the hopes that his packets make it thru. This is the classic ALOHA network configuration with "hidden terminals". It can be shown that collisions on the uplink channel will statistically reduce the channel capacity so that only (1/2e) = 18.4% of the packets make it thru. Thus, the downlink (on which there are no collisions) can support about 5 times as much traffic as can a single, collision-limited uplink.

There are two ways out of this dilemma. First, the uplink users could use a data rate about 5 times the downlink; this approach was taken by the AMSAT-DL designers of AO-13's RUDAK experiment where a 2400 bit per second (BPS) uplink is balanced against a 400 BPS downlink.

The second approach is to have multiple, separate uplink receivers. The FO-12 satellite has four 1200 BPS uplink channels balancing one 1200 BPS downlink.

MODEMS AND RADIOS FOR PACSAT

For our PACSATs, we have allowed for both solutions to the ALOHA limit. Like FO-12, there are to be four user uplink channels, however each of which can be commanded to support 1200, 2400, 4800 and possibly 9600 BPS uplinks. The downlink transmitter will start its life at 1200 BPS, but higher rates should be possible.

Our design was heavily influenced by a decision we made early on: we would only use standards which were supported and available "off the shelf". Thus when our PACSAT comes to life, the ground user can use the identical hardware he uses for FO-12 today. The user's uplink will be at 1200 BPS, Manchester-encoded FSK and the downlink will be 1200 BPS binary PSK. These standards are supported by the TAPR and G3RUH modems, by the myriad FO-12 modems available on Akihabara in Japan, and by the DSP modems that N4HY and I have been working on.

These "mo" modulator in these modems plugs into the mike jack on a stock 2M FM radio, which we assume can be tuned in 5 kHz steps. The satellite link margins should be such that 10-25 watts into an omnidirectional antenna should be adequate (providing everyone runs similar power).

The "dem" demodulator plugs into an SSB-capable 70 cm receiver or all-mode transceiver, which needs to be tunable in 100 Hz (or preferably finer) steps. The PSK downlink should be "Q5" even with an omnidirectional antenna, providing the local noise level is low.

The spacecraft's receiver has 15 kHz wide channels, regardless of the bit rate programmed at the spacecraft. The 1200 BPS data rate combined with an FM deviation of < 3 kHz, plus doppler shift, plus 5 kHz steps on a typical FM radio just fit the 15 kHz bandwidth. At some later date we will begin enabling selected uplink receiver channels for higher data rates (like 4800 BPS), but the user will now have to pre-steer the doppler and set his frequency more accurately than 5 kHz. Also most stock FM radios will not pass the 4800 BPS data rates without significant modifications.

ONBOARD PACSAT

Let us now discuss some of the features of the satellite's architecture. The electronics is divided into modules, with the space inside each module being about 7.8" x 6.5" x 1.5". The mechanical layout has five of these modules stacked atop each other as shown in Figure 2, which we will describe from top to bottom.

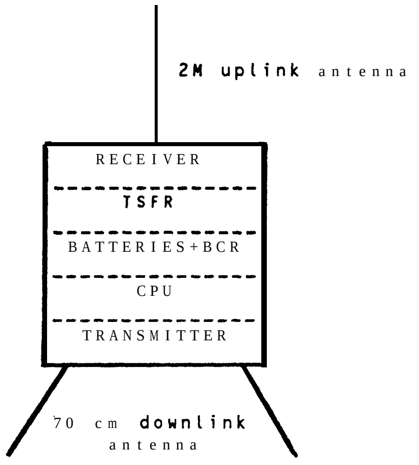


Figure 2. PACSAT LAYOUT

RECEIVER

The core of the receiver is the Motorola MC3362 single-chip FM receiver, couple with a stock NDK crystal filter with 15 kHz bandwidth centered at 10.7 MHz. The filter has very good skirts, with 80-90 dB ultimate rejection. The input to the 3362 is an IF in the 40-50 MHz range. The 1st LO in the 3362 is crystal controlled to mix to 10.7 MHz. Following the filter, the 3362's second mixer is driven from a crystal controlled 8.9 MHz 2nd LO to produce a final IF of 1.8 MHz selected for best linearity of the MC3362's FM detector (discriminator).

The MC3362's FM detector drives two matched data filters, each of which uses one section of a TLC274 CMOS op amp; the 2-pole Butterworth filters are optimized for 1200 and 4800 BPS data rates. A CD4066 analog switch selects the output of one of the two filters to drive the data clipper section of the 3362. The appropriate filter is selected by the CPU.

In addition, one section of the TLC274 produces an analog signal in the Ø-2.5v range corresponding to the user's frequency (the "disc meter") and another produces a Ø-2.5v analog signal corresponding to the user's signal strength (the "S meter").

All this circuitry takes up 1.5" x 3" on the receiver's circuit board and draws under 20 mW (< 4 ma at 5V). This circuit is replicated five times to provide the 4 user uplink channels plus a command/control channel.

The design of this portion of the receiver was done by W3IWI with invaluable inputs from Eric Gustavson (N7CL).

In front of this bank of five FM IF strips is a fairly conventional GaAsFET preamp with a noise figure < 1 dB. A narrow-band 3-stage helical -filter provides selectivity between the GaAsFET preamp and a dual-gate MOSFET mixer which is driven by a crystal-controlled LO at about 100 MHz. The output of the MOSFET (at 40-50 MHz) drives five emitter followers to provide isolation between the five FM IF stages. The design of these stages was done by Jim Vogler (WA7CJO) and W3IWI.

The total power consumption for the entire receiver is about 150 mW.

[As a side note -- the receiver modules designed for PACSAT have been made easily reproducable, with very few "twiddles". All components, including the coils and helical filter are off-the-shelf' items purchasable from sources like Digi-Key. It is anticipated that TAPR and/or AMSAT will make single-channel receiver kits available for use in dedicated packet link applications if there is enough interest].

TSFR

For PACSAT, this is a. dummy module. TSFR means "this space for rent", and is reserved for future expansion.

POWER SYSTEM

The Battery Charge Regulator (BCR) module contains the NiCd battery pack, the charger that conditions solar panel power to charge the batteries, and the switching regulators that produce the +5 and +10 v power needed by each module. The BCR and regulator design was done by Jon Bloom (KE3Z) with help from Gordon Hardman (KE3D).

The solar panels make use of high-efficiency silicon cells with back-surface reflectors (BSR). BSR technology is new, but it allows for much higher efficiency; if a photon does not produce electricity as it passes thru the silicon on its way in, the reflector allows a second chance to "grab" it. The solar panel electrical and mechanical design was done by Jan King (W3GEY) and Dick Jansson (WD4FAB), and the solar panels are being produced under contract by Solarex.

The price of space qualified NiCd batteries has become prohibitive,, so new, low cost approaches have been adopted. Larry Kayser (VE3PAZ) and his group in Ottawa proved with UO-11 that if good, commercial grade batteries were purchased, they could be flight qualified. The qualification procedure involves extensive cycling to characterize the charge-discharge curve and temperature performance, X-raying the batteries to look for internal structural flaws, then selecting only the best cells, and then finally potting the batteries.

While the solar panels produce about 14 watts, when averaged over a whole orbit (some time is spent in eclipse), and after losses in power conditioning about 7-10 watts is available.

CPU

In many ways the flight computer is the key to PACSAT. At the time we were selecting the CPU, the SANDPAC group in San Diego were finishing the first pre-production run of the new PS186 network switch. Based on their experience, we selected a similar architecture. The flight CPU is based on the NEC CMOS V-40 CPU (quite similar to an 80C188). The flight CPU includes EDAC (Error Detection and Correction) memory for storage of critical software, plus bank-switched memory for data storage (i.e. "RAM Disk"). We hope to fly upwards of 10 Mbytes on each PACSAT (limited only by available space and the price of memory chips). The CPU, when running hard draws about 2 watts of power.

A companion paper by Lyle Johnson **(WA7GXD)** and Chuck Green **(NØADI)** describes the CPU's architecture in much more detail. A paper by Bob **McGwier (N4HY)** and Harold Price **(NK6K)** describes the multi-tasking software. Jim **DeArras (WA4ONG)** is converting Lyle and Chuck's wire-wrapped prototype to multi-layer circuit board. The ROM-based bootloader to allow recovery from disasters has been written **by** Hugh Pett **(VE3FLL)** whose code had previously saved the day on **UoSAT.**

TRANSMITTER

At the time of this writing, the transmitter is still in the design phase, so some of these parameters **may** change. The transmitter will be BPSK modulated, and will have its power output changeable by ground command. The current plans are for two power levels, about I.5 or 4 watts. The transmitter starts out with a crystal oscillator at 109 MHz, and is followed by two doublers to 436 MHz. This design is being done by Stan Sjol **(WØKD).** Gordon **Hardman (KE3Z)** is working on a power amplifier using a Motorola **MRF750** driver and a MRF752 output stage. The collector voltage on the driver stage will be command selected to be either the **+5** or **+10v** bus to provide power agility. This collector voltage may be amplitude modulated to provide some time-domain shaping to minimize the transmitted bandwidth. Transmitter development is also being done in Canada by Bob Pepper **(VE2AO).**

GLUE

The myriad mechanical details were all sorted out before we cut a single piece of metal by Dick Janssen **(WD4FAB);** Dick made extensive use of modern CAD techniques and all drawings were done with **AutoCAD** (see Figure 3). In Boulder, Jeff Zerr has been shepherding the detailed mechanical layout and find what pieces don't fit. A "show and tell" model was built by **????** with help from Dick Daniels, and a mechanical mockup for vibration testing has been built by Jeff Zerr.

When we began developing the **Microsat** concept, we took a look at problems that had been major hassles on earlier satellites. High on the list were problems in building a wiring harness and testing individual modules. We also wanted a design that allowed a "cookie cutter" approach to manufacturing since **we** anticipate a number of launches in the **next** few years. We came to the conclusion that we needed to develop a bus-like wiring approach with all modules having similar interfaces, and we needed to minimize the number of wires. I took on the task of solving this problem and defining the electrical "glue" that holds the system together.

After exploring a number of options, the design we adopted was to use hi-rel DB25 **25-**pin connectors on each module and use a **25-**wire bus made like a flexible printed circuit. Of the 25 wires, about 40% are used for power distribution, about 40% to carry packet data from the receiver to the CPU and from the CPU to the transmitter, and the **final** 5 wires are used to let the CPU control functions in the individual modules and for analog telemetry.

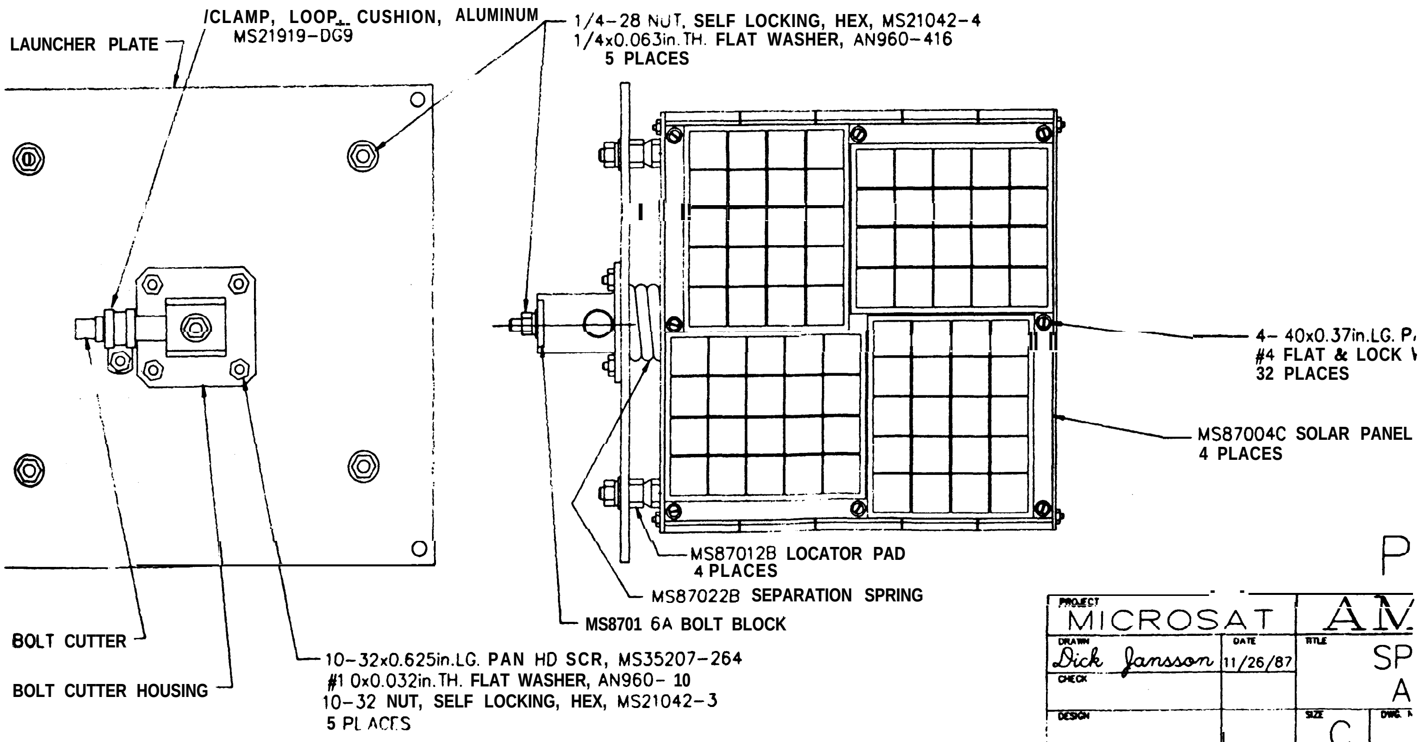


Figure 3. Part of one of WD4FAB's drawings showing UICROSAT assembly details.

AART

In order to squeeze all these command, control and telemetry functions into only five wires, we have built a very small (7 inches long!) LAN with the CPU acting as the network master node and each module being a slave node. Data communications from CPU to module consist of two byte packets; the first byte (with the MSB=1) addresses up to 128 slaves, and the second byte (with MSB=Ø) is a 7-bit received data field to be passed to the module (RXD). On receipt of a valid address, the module automatically sends back two 8-bit bytes (TXD) of data on another wire. All data is sent with normal asynchronous protocols.

On the CPU side, this async data is generated and received by the UART built into the V40 chip. The protocol is easily simulated on a PC, so testing each module does not require a complete working spacecraft.

In each module, we use a clever IC: the Motorola MC14469F Addressable Asynchronous Receiver/Transmitter (AART). The 14469 is a 44-pin surface mount part (also available as a 40-pin conventional DIP) which implements the protocol just described with very few external parts. It has separate pins for the 7 address bits, the 7 RXD bits and the 16 TXD bits.

The 7 RXD bits are used for a number of functions. The MSB of this word is used to select analog vs. digital functions, with the control data specified by the remaining 6 bits. For digital functions, the 6 bits are treated as two 3-bit nibbles which constitute the address and data for three CD4099 addressable latches, resulting in 24 bits of digital data being available for control functions in the module.

When the MSB selects analog functions, the 6 bits are taken as addresses for CD4052 CMOS analog multiplexer chips which decode 6 discrete analog telemetry samples plus four thermistors. When a module is selected in analog mode, the selected analog signal is switched onto two wires (signal plus return) in the 25-wire bus, and when the module is de-selected the two wires are floated. A single, fast 8-bit Ø-2.5v A/D converter in the CPU handles all spacecraft analog telemetry. Each module is responsible for preconditioning its analog signals to fit the 0-2.5v range.

All these parts, including some op amps to condition the thermistor signals, plus the DB25 spacecraft bus interface connector and tie-points for all signals needed in the modules are fitted onto a 7.8" x 1.5" board which is mounted against one wall of the module frame. The interface boards in each of the "slave" modules are identical except that the AART chipp 'is strapped to different addresses. This small board has been dubbed the AART board. It was designed by W3IWI and Bob Stricklin (N5BRG). Each board requires 5 mW of power (about 1 ma at 5v).

THE OTHER MICROSATS

DOVE

So far we have described the two Microsat PACSATs: those sponsored by AMSAT-NA and AMSAT-LU. The BRAMSAT DOVE spacecraft is still in the final design phases, but it will be built from many of the same pieces and will have the same general mechanical layout. DOVE will transmit its digitized voice signals in the 2M band with conventional FM modulation. Rather than designing a different receiver system, we have decided to have the command uplinks also on 2M; the DOVE transmitter will turn itself off every few minutes to listen for commands. Only the transmitter module is different for DOVE. As of the time of this writing we are planning to use differentially-encoded voice synthesis (e.g. "delta modulation'*) with up to 4-bit encoding of the differential data. Preliminary design on the speech synthesizer has been done by Bob McGwier (N4Y) and W3IWI and is being simulated using our DSP hardware.

NUSAT

The Weber State NUSAT MICROSAT is different mechanically from the PACSATs, shown in Figure 4.

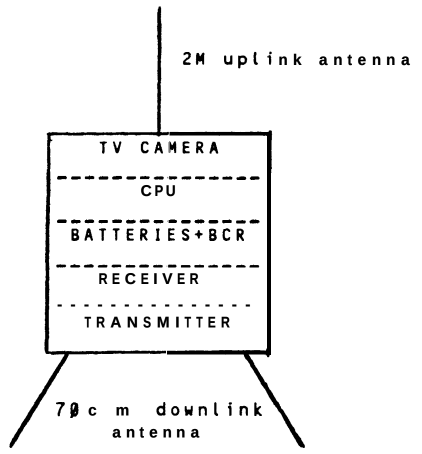


Figure 4. NUSAT LAYOUT

The major difference is that NUSAT has a CCD TV camera in the top module. The TV camera is connected to a high-speed multi-channel "flash" A/D converter which can digitize incoming video signals at 10 MHz sample rate. Its data is stored in memory which can also be accessed by the CPU. The Weber TV camera module and CPU were placed in adjacent modules so that the memory could be easily dual-ported.

The sample rate for the A/D converter and the input signal source can be selected by the CPU. The primary signal source is a CCD TV camera equipped with an electromechanical iris built into its lens. The iris's aperture can also be controlled by the CPU. The camera's field of view allows a 350 km square to be imaged from the satellite's 800 km high orbit.
The camera assembly occupies about 1/4 of the space in the module. It is planned to use video data compression techniques to minimize the downlink data requirements; Weber State and AMSAT-NA plan to have software to support these advanced video techniques available around launch time.

Weber State also plans to try a 1269 MHz video uplink. Video data from this uplink will be digitized by the "flash" A/D converter and loaded into the dual ported memory, just like data from the CCD camera. It is also hoped that the TV camera can be used as an visible and IR spectrometer covering the 400 to 2000 micrometer wavelength band.

The other NUSAT modules are nearly identical to the PACSATs and NUSAT could be also turned into a PACSAT merely by loading different software.

The Weber State team consists of a number of students, staff and faculty members from the Center for Aerospace Technology (CAST) including Bob Twiggs, Bob Summers and Chris Williams.

THE FIRST MICROSAT LAUNCH

AMSAT-NA and the UoSAT group have worked with the European Space Agency and Ariannespace to develop a new launch capability for very small satellites. This will be first tested on the launch of the SPOT-2 Earth Resources Satellite in early 1989. On that flight there will be SIX small satellites -- our four Microsats and two somewhat larger UoSAT spacecraft. The orbit is nearly ideal -- sun synchronous at 800 km altitude, much like the Oscar-8 orbit. At mid-latitudes, passes will occur twice per day at predictable times around 10:30 A.M. and 10:30 P.M. local time.

USING THE MICROSAT SATELLITES

As we mentioned before, our PACSATs and Weber State's NUSAT use ordinary AX.25 packet protocols. To receive any of the three, you merely need to add a PSK demodulator to your 70 cm receiver. The uplink requirements are modest and the same as FO-12. At a later time, when transmitter technology permits and user loading dictates, some of the receiver channels will be reprogrammed to higher speeds. But initially, if you are able to use the FO-12 satellite, then you are all set.

The spacecraft software that you will see will be designed for message handling, and the code is being written by Bob McGwier (N4HY) with inputs from a number of us. The initial software will probably look very much like a W0RLI/WA7MBL BBS system, with a few enhancements. First of all, the prompt that the satellite will send to you will have two telemetry numbers in it -- these are your signal strength and discriminator meter readings. The discriminator meter should be invaluable in helping you center your signal in the receiver's passband and its use will become mandatory as we migrate to higher uplink speeds. The spacecraft software will support multiple, simultaneous users. There may be commands that allow you to request specific telemetry information from the satellite.

I anticipate that much of the utility of these satellites will be as an augmentation of the terrestrial HF long-haul message forwarding networks. If this proves to be true, then fully automated gateway stations will make heavy use of the satellite capabilities.

Therefore it is important that we design both the ground-based and flight software to work together smoothly. We have had ongoing discussions with the writers of BBS code (like WORLI and WA7MBL) to make sure that both sides of the link will be ready on launch day. In these discussions we have been devising schemes so that the burden of maintaining routing information resides on the ground. New forwarding protocols in which the receiving station tells the sender what message addresses it can handle are being defined. It is likely that these will be coupled with heirarchial domain-oriented addressing schemes like are used by TCP/IP protocols. A user on the W3IWI BBS would have an address like W3XYZ@W3IWI.MD.USA and if I were operating as a gateway for the MD/VA/DE/PA/NJ area, I would be able to inform the spacecraft to send me any messages so addressed.

At the same time that "connected" mode activity is going on, the satellite will be sending UI "broadcast" (i.e. UNPROTO) frames with telemetry and bulletins of interest to all. On NUSAT, digitally encoded pictures of the earth will be sent as UI frames which will be reassembled by the user on the ground.

THE FUTURE

We have reason to believe that there are a number of launch opportunities to LEO for very small satellites. We have designed our Microsats to be easily reproducable. As new capabilities (perhaps 9600 or 19,200 BPS modems? Experiments to fit into the TSFR module?) are developed, we feel there will be opportunities to fly them.

We anticipate non-amateur uses of our technology. Initial discussions with scientists specializing in oceanography and seismology have shown that they have a need for low-cost data collection systems from remote locations. We anticipate a scheme for a commercial licensee to "sell" our technology in these markets. Just like royalties from TAPR's TNC2 project have provided resources for future development activities in packet radio, we hope that Microsat royal ties will provide a similar legacy for advancing amateur satellite technology.

We also see that the Microsat technology provides a perfect way for fledgling space groups associated with other AMSAT organizations around the world and with universities to develop their own satellite programs. Don't be surprised to see Microsats being built by people from many nations.

The spacecraft operating software can be uploaded from the ground. As NK6K and N4HY discuss in their companion paper, the software we will be flying is the most complex ever attempted in the amateur satellite program. It probably will crash! We have designed in several safeguards to make this possible. With this flexibility, we also have the ability to try new things. Perhaps we will see new mail-handling protocols developed which use datagrams. Perhaps we will see a PACSAT programmed to be a TCP/IP FTP file server. As the old adage states:

IT'S ONLY SOFTWARE !

PARTING COMMENTS AND ACKNOWLEDGMENTS

The most important "glue". that holds a project like this together is the project manager. We are indeed fortunate to have Jan King (W3GEY), with his wealth of experience, his contacts in the aerospace industry, his mother-hen persistence in reminding us of the rigors of space, and his compulsive personality to make sure everything happens.

Jan's "glue" binds together a team of high-strung, emotional prima donnas who are equally compulsive. Many of the team members have invested a lot of 3AM mornings working on this project! All the team members have had to wear very thick skins to withstand the FLAME ON! communications blasts some of us are prone to 'emit. Bob Mcgwier, Dick Jansson and Lyle Johnson all deserve special credit for service above and beyond the call of duty.

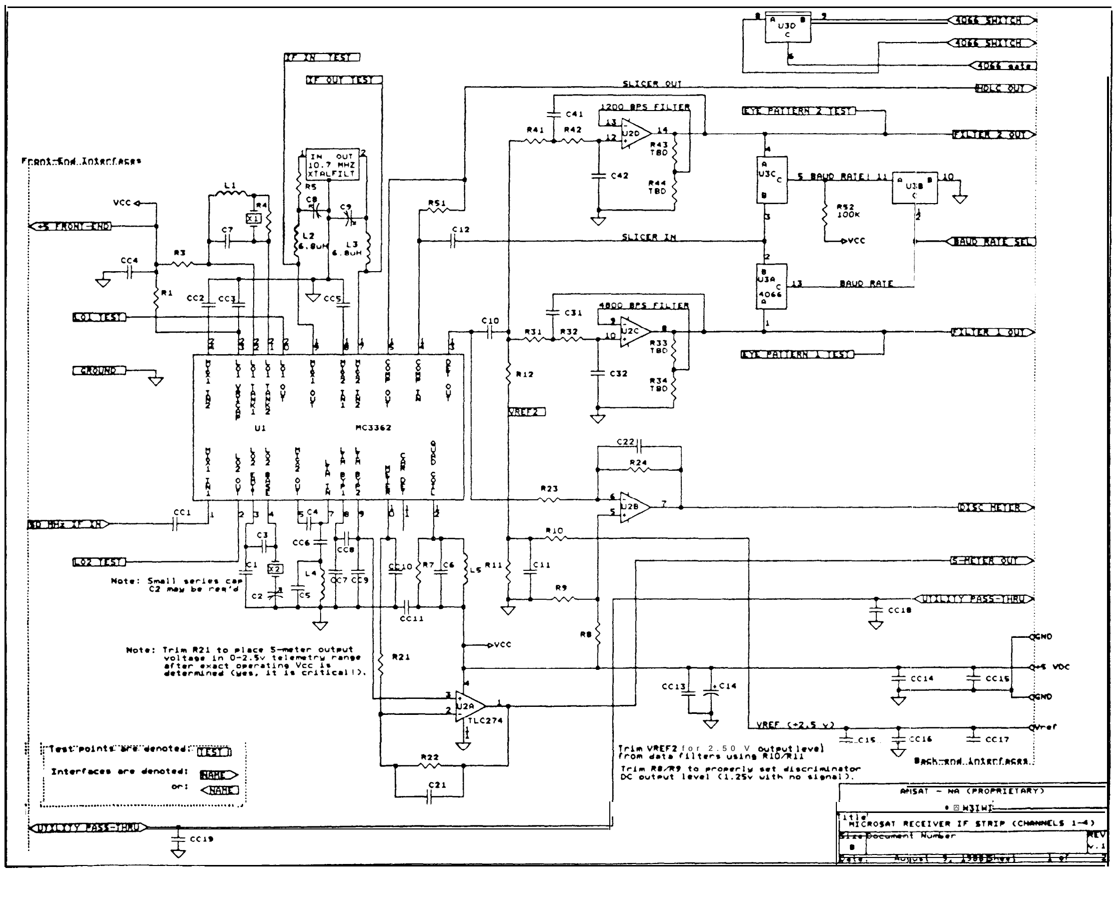
This project has significant players spread out all over North America, with major activities in NJ, MD, VA, FL, CO, UT, AZ, TX and CA. Unfortunately amateur radio communications are inadequate to keep such a dispersed team working together. We have relied heavily on commercial electronic communication channels, particularly AMSAT's network on GTE TeleMail and TAPR's channels on CompuServe, plus a lot of phone calls. Every few months we get a number of the people in one place and lock the door to make sure everyone REALLY understands what is happening.

We have made heavy use of various CAD tools during the development activities. Mechanical layout was done with AutoCAD. ORCAD . was heavily used for developing schematics, wiring lists, parts lists and net lists. CAD PCB layout used Smartwork, ORCAD PCB and Tango. See Figure 5 for an axample of some of this use of CAD techniques.

We have done some experimentation using higher-level networking for technical communications to move CAD data using my TOMCAT FTP file server which has a SLIP port in addition to being on the "real" network.

There are two organizations not mentioned earlier that have contributed a lot to this project: TAPR and the ARRL. For many of the volunteers working on this project the distinction between TAPR and AMSAT is' fuzzy since they seem to wear two hats. In addition to the TAPRites working on this project, TAPR has made vital contributions of funds and hardware, without which we couldn't make it. Special thanks to Andy Freeborn (NOCCZ) for helping to make the TAPR/AMSAT interface smooth. At the ARRL labs in Newington, Paul Rinaldo and Jon Bloom have made many vital contributions.

From the AMSAT organization, two people deserve a lot of credit. Vern Riportella (WA2LQQ) was instrumental in arranging the AMSAT-LU and BRAMSAT participation in the project. Martha Saragovitz has acted as mother confessor, paid bills, handled meeting logistics and kept smiling thru it all, despite repeatedly crying out "Where's the money coming from?".



IF IN TEST
IF OUT TEST
SLICER OUT
1200 BPS FILTER
4800 BPS FILTER
EYE PATTERN 2 TEST
EYE PATTERN 1 TEST
FILTER 2 OUT
FILTER 1 OUT
HDLC OUT
BAUD RATE SEL
DISC METER
S-METER OUT
UTILITY PASS-THRU
+5 FRONT-END
LO1 TEST
LO2 TEST
GROUND
10 MHz IF IN
MC3362
TLC274
4066
VCC
Note: Small series cap C2 may be req'd
Note: Trim R21 to place S-meter output voltage in 0-2.5V telemetry range after exact operating Vcc is determined (yes, it is critical!).
Test points are denoted: TEST
Interfaces are denoted: NAME
or: NAME
Trim VREF2 for 2.50 V output level from data filters using R10/R11
Trim R8/R9 to properly set discriminator DC output level (1.25V with no signal).
VREF (+2.5 V)
AMSAT - NA (PROPRIETARY)
W3IWI
MICROSAT RECEIVER IF STRIP (CHANNELS 1-4)

Figure 5. One of W3IWI's ORCAD schematics of the MICROSAT receiver IF strip.

The DSP Project Update

by
Dr. Thomas Clark, W3IWI and Dr. Robert McGwier, N4HY

1. Introduction

TAPR and AMSAT have signed a formal agreement which forms a joint project. The purpose is to bring the rapidly advancing technology and techniques of digital signal processing to bear on the communication needs of amateur radio. The AMSAT-TAPR digital signal processing project has made steady progress over the past year on both software and hardware. Lyle Johnson is leading the hardware effort in Tuscon and will report on that progress elsewhere in these proceedings. We will report on work that is ongoing to choose what the second generation DSP unit will look like.

The project has been very lucky to have several new people come along and offer valuable suggestions and in a few cases some new software. Motorola, Inc. and A.T.&T., Inc. have donated development systems, none of which we have running at the time this is being written but all within days of being handed to the project. The project has acquired the use of complete development system and software for the TMS320C25, T.I.'s current best that is available for OEM's 'over the counter.' That acquisition is a story that will amuse you and that will be covered. The TAPR-AMSAT project, whose current planning and future goals you will be told about here, believes strongly that we are leading amateur radio into new possibilities for special purpose communications modes being as easy as RTTY and packet and thus involving many more people in these areas of our fascinating hobby. Indeed in some cases, this could raise the involvement in many of these speciality modes significantly.

2. Software Developments

At the last networking conference, we reported on the spectrum analyzers and a modem for the reception of JAS-1 or FUJI OSCAR 12 and about ongoing software work at that time (1)(2). The work continues to emphasize modems and applications, of fast fourier transforms which are at the heart of our spectrum analysis tools. All software efforts in the past year have been on either the DSP-10 boards the project acquired from Delanco Spry (3) or on the PC-clones in which they run. The FUJI Oscar 12 modem took on a new guise which has been very effective in demonstrating the functional versatility of DSP approaches to communications systems. We decided to add a BEL-202 modulator following the 1200 bps PSK demodulator. This enables one to copy the signals from JAS-1 with the DSP board and an unmodified tnc. We will be using this piece of software a great deal in the coming months after it has been modified to run on a dedicated DSP box the project will produce for the benefit of amateur radio. In the early part of next year, PACSAT 1,2, and 3 will be launched. PACSAT 3 will have a CCD camera on board as its primary mission. All of these satellites use PSK on the onboard transmitter at rates between 1200 bps and 4800 bps. The DSP software will support these satellites as the demodulators. The uplinks are Manchestered FSK with the rates as before. This is also easy to produce and will be part of the software work that will be available for distribution with the initial units.

N4HY also has working a Hilbert transform demodulator, of the type mentioned in last years proceedings, in DSP56001 'C' code and less optimal versions in TMS32010 code (due to smaller data space) which demodulate QPSK at rates up to 9600 BPS, BPSK to 4800 bps, coherent demod for GMSK to 4800 bps all with the same carrier and data extraction loop. The only change is the input of the minimum allowable phase transition during bauding. The data decision algorithms differ considerably from one type modem to the next but the same basic element is common to all these modems. In some cases, filter coefficients have to be changed (change a couple of dozen numbers in the data) to allow for different width spectra, etc. The common thread in the demodulation of phase modulated data signals is never more apparent than when the demodulators are written in software. The DSP56001 is fully capable of going faster than the numbers mention above, but these are the upper limits due to the TMS32010.

N4HY and KA9Q worked on DSP and PC software that does OSCAR 13 PSK telemetry modem functions and also does decode of the telemetry frames from the spacecraft. This was used to monitor tank pressures, battery voltages, and many other critical values relating to the health and welfare of the spacecraft during the critical engineering phase immediately following launch. The modem works well to small signal to noise ratios now but could use some more optimization of the parameters which control the phase locked loop which is at the heart of the system. Both the Oscar 12 and 13 modems are based on the Costas Loop demodulator (4). The work remaining to be done is the optimization of the numbers in the program which govern the dynamical behavior of the 'loop filter' which determines its response to various phase and frequency error it is trying to eliminate. This should allow the DSP modems to work down to smaller SNR in the case of Oscar 13. The dynamical behavior of the loop needs to be different in the case of high doppler and high SNR such as will be the case in the PACSAT's to be launched next year (5)(6)(7).

Two approaches to be1202 demodulators have been attempted. One is a PLL used as an FM demodulator and the other is a filter discriminator modem. The first attempts to track the frequency variation as the signalling changes from one tone to the other. The latter divides the passband into two parts by using a pair of filters. Bit decisions are based upon which filter shows a higher energy content. The former allows for much better data carrier detect to be built and the latter is easier to modify. It can't be too much longer that we will have to deal with these modems as the DSP box and better hardware modems come along to replace them. Nevertheless it is of at least academic interest to determine which of the approaches AFSK can be best done in an inexpensive DSP chip.

One of the most powerful tools that DSP allows us to make is a spectrum analyzer. This has wide applicability to many areas. W5SXD has given us a very nicc EGA display for one of our early fast fourier transform routines. John Connor, WDOFHG has produced a useful DSPSCOPE utility for doing an audio oscilloscope function and it works nicely for examing waveforms. Franklin Antonio, N6NKF probably takes the prize for stirring up the troops as much as anyone. He wrote a routine, based upon the work of Burrus and Parks that runs in the PC along with the display routines. It did a creditable job of running at close to the same speed as one done by the authors of this paper and he uses the Delanco Spry board only for sampling. It is a smaller FFT but the display was autoselect EGA, CGA and could be mouse driven. Investigation by N4HY revealed that Burrus-Parks had the faster FFT's already done in TMS32010 code. They needed quite a bit of changing to run on the different architecture of the DSP-10 but they have allowed real valued signals to be done with one-half size complex FFT's saving time and storage. They are also much faster as they are radix four and radix 8 FFT's with three and two butterfly's respectively. Then a partial butterfly is needed to take the half size complex FFT and get the real signal FFT output from it. This gives us a much more serious tool for spectrum analysis of audio signals. (These are 'real' signals. Those that come out of your speaker are real valued signals). The code that runs on the PC is being optimized to run in hand coded assembler in critical areas where the speed bottleneck now exists. Linear and Power displays are now working along with a variety of other functions. This is a great deal faster than the old routines.

We are also studying in considerable detail, the optimal statistics to be used in weak signal detection problems. W3IWI did a detailed looked at the statistics of the signal returned from the moon. This study along with more recent input from Dick Goldstein of JPL, and Vince Poor (8) of University of Illinois also gave us some detailed input about this problem. 'With the recent success of I2KBD, DSP team member from Italy, in receiving his own echoes from the moon, we are again spurred on to applying this work to the task of truly QRP EME. This technique will not and should not be limited to QRP EME. It can and probably will perform much better along weak signal paths terrestially. It should give VHF-UHF types a tool that will allow QSO's using signals that were previously unusable. The coherence time should be much longer than those returned from the moon and the real power of the FFT will shine in these cases. In the early experiments done by the authors over a 150 mile path on 78 cm with satellite arrays were astounding to say the least. The power could not be turned low enough at N4HY for W3IWI to prevent detection on the screen using a crude display program. This can and should be revisited soon.

Another application of the F'FT based spectrum analyzer has been occupying the thoughts of VE3JF, N6NKF, W3IWI, and N4HY. 0 ne of the great needs in packet radio and HF digital modes in general is a better modem for HF. The general idea is to send more than one tone at a time and encoding more than one bit of data during each baud or signal element. This will alleviate a large part of the damage HF propagation does to a digital signal. The multipath problem causes each signalling element to arrive from a random number of reflectors and all arriving at different times. This can cause destructive interference. If you signal at a slower rate, the likelihood of this multipath distortion causing you to make a decision error goes down considerably. If we encode 6 bits in each signal element and then transmit this at 50 signal elements per second (50 baud), we will be transmitting data at 300 bits per second. Several studies have been done which show that somewhere near 50 baud is nearly optimal for HF digital transmission in the 40 meter and 20 meter bands. This is one of the reasons AMTOR is so successful at getting the data through (along with its relentless retransmission). We need to spend a great deal more time developing this capability as it will be of great benefit to HF digital transmission modes.

N4HY has considered several. specialty modes over the last year. The first to receive a concerted effort has been WEFAX-APT. This is the Automatic Picture Transmission from the NOAA weather satellites. For several years, many reference books (Taggart from 73 and ARRL Handbook) have claimed that a phase locked loop is not the best way to go in copying weather picture transmissions from the low earth orbitting weather satellites. The scheme used by both NOAA (USA) and Meteor (USSR) satellites is to encode the picture element at the current time as the amplitude of a pure tone. For the NOAA birds, this is very close to a 2400 Hz tone, sent into a wideband (70Khz) FM transmitter. When the amplitude is near minimum, the picture is black. When the amplitude is maximum the pixel is white. The same is true for the Soviet satellites with the exception that there is quite a variance in the frequency of the tone from one satellite to another. Most PLL's used in the demodulation of these satellite signals are first order PLL's with very narrow bandwidths in order that recovery of the signal may be done in noise. It would appear that most of them are tuned for the nominal frequency of 2400 Hz used by NOAA. When the Meteor satellites, with their low modulation index for black, are used with these 'mistuned' first order PLL's they do not function properly and the results are poor. Here is a strong case example for using DSP. Nothing had to be changed in the program to make it work PERFECTLY with the Meteor satellites except the frequency. Upon changing one number in the DSP program, the demodulator worked perfectly and the raw pictures one gets with the Meteor satellites are preferable to NOAA pictures since the dynamic range is smaller in the NOAA pictures. This may no longer be true after we begin working on image enhancement for these pictures. The conclusion is that the Meteor signals are phase continuous during the picture even when the modulation falls to small amplitude. A first order PLL will just receive a small error signal during this period and will not adjust the phase. If you have a small frequency error the PLL will just spin through these areas of low amplitude modulation and come out of them with small phase errors at most. This procedure yields several dB of signal to noise ratio improvement over the product detector approach used in so many WEFAX-APT demodulators.

3. **Hardware donated, Future DSP projects**

The major DSP chip manufacturers have both wittingly and unwittingly been of great aid to the DSP project. Motorola donated two DSP56001's, a boot ROM which includes a monitor, and bare boards for development purposes to our project. The DSP56001 executes 10.25 million operations per second and this number is an underestimate of its real capabilities. The reason is that like the T.I. chip we have been using, it does in parallel those things which make DSP very efficient on these chips. However, of all the second generation chips we have evaluated to date, this is clearly the most capable. Even without careful coding we should be able to run 19.2 kbps without much problem doing both FSK and PSK modems of several different varieties. Lyle Johnson, WA7GXD, and the authors have considered what should be the next step after the DSP-1, which is considered in these proceedings. We believe that we should have a very capable DSP board that will carry the DSP56001 at its heart, which will be available as a replacement card for the TMS32010 board in the DSP-1. Steve Sagerian, KAOYRE, is building the Motorola donated kits for the project. Serious coding will begin on this card once this construction is complete. Steve was also responsible for securing the donation for the project.

NN2Z, Dave Truly, has become well known to many tcp-ip enthusiasts as the current author and manager of the bm mailer program in KA9Q's, net.exe program. Recently, Dave was asked to join the DSP microprocessor support group at ATT. Shortly after joining this group, Dave began pushing the concept of this group donating some hardware and software to the DSP project. The group leader has decided to do exactly that. The DSP-32 development engine with the support software will be donated to the project. Dave is learning as much DSP as he can and will also be working on code for the DSP-32 with N4HY in support of our evaluating the DSP-32. We believe that this will be a useful product for the project to consider constructing.

Companies can sometimes even make unwitting contributions to our efforts. We don't turn them down just because they didn't intend for them to be of benefit! N4HY works at IDA/CRD in Princeton, NJ. He works in the area of signal detection and estimation. Also working there is Maureen Quirk, Ph. D. an engineer specializing in DSP and signals. They are partners on many technical projects. During the spring of this year, N4HY had to go to England for work at the time of the largest gathering of people specializing in signal and speech processing, the annual international meeting sponsored by the Acoustics, Speech, and Signal Processing section of the IEEE. Maureen decided to go to all vendors and gather as much information on DSP chips and the latest products for the benefit of N4HY. While doing this favor, she had a card from T.I. stamped at each vendor of a T.I. DSP chip product. Returning this card enabled her to participate in the drawing for the grand prize. She won. She got the ASDS from T.I. This is the complete software development PC plug in card for the TMS320C25. The memory chips alone, 128K of 25 nanosecond static memory, are pure gold even if they are silicon. She doesn't own a PC. Thinking that she would never use it, she started to give it back. Realizing that their would be death and destruction upon her return and her usual partner finding out about this curious circumstance, she decided in favor of life. The board now has a permanent home in the 386 machine residing at the QTH of N4HY. But of course, Maureen may use it ANY time she wants to!

We will be evaluating these DSP chips for the TAPR AMSAT DSP project and studying the best way to make use of all these resources. With all these development tools at our disposal, we will be able to give a fair evaluation to each of these chips and find the strength and weaknesses of each. This will be reported in the next Networking conference.

4. Future, **Pie in the Sky, etc.**

If the future of the DSP-1 is as bright as we believe it will be, one of the great benefits such an engine could provide would be as an educational tool. It would be an inexpensive approach to having several different DSP chips along with a host processor to interface with it available on an open architecture for programming. This is ideal for amateur radio experimentation and for educational purposes which are not easily seperable in this case. The current costs for one of these development engines for each of the DSP chips is several thousand dollars. For about the same cost one could have a less capable development system but be able to put many different chip/boards in it for evaluation and comparison. The obvious value to education does not need to be belabored further. This particular project has an audience that greatly exceeds amateur radio as some others have in the past. We are wiser than we have been in the past and are looking at this project as a means to do future projects. The interest in the outcome of this project is demonstrated by major companies *donating* hardware and software valued at several thousands of dollars. The applications are limited by people's imagination and time more than any other factors.

As evidence of growing interest, the IEEE ICASSP meeting for the coming year is in Glasgow, Scotland. This is the next meeting of the group where the TMS320C25 board was won. The authors of this paper have had accepted a paper on the DSP-1 and our project and we will represent our group at this conference next May.

5. People and Conclusions

The people in this project are what it is really about. The array of talents allied against the communications needs of amateur radio is very impressive. We need to thank Courtney, N5BF for beginning to try and give a responsive central location for information on DSP. Paul, KB5MU and Courtney have done a very nice job of producing a schematic of the Delanco board for the internal use of the project members. It has been. invaluable aid in deciding what was done wrong and what was done right. John, WD0FHG deserves a great deal of thanks for his mailings of the DSP diskettes. TAPR is also mighty lucky to have Andy Freeborn, NOCCZ to be president. The value of his aid in managing the DSP project administratively cannot be overestimated. By next year, we should be here telling about all the neat stuff that runs on the DSP-1. We hope several of you will have it in your shack by conference time in 1989. For those who have despaired of progress, we hope that this DSP year in review has enlightened you. We also remind as often as we can, look at the people who are writing the papers on DSP and on PACSAT. The upcoming launch date early next year will not wait while DSP can, no matter how loathsome the prospect is. Whenever you write a report on a years activities in a group as diverse and widespread as this is, you inevitably leave some contribution out that should have been described. If we have made such a faux pas, it was inadvertent. For those of you with access to ARPANET, you may find the entire contents of the DSP mail distribution available in several diskette images. FTP to tomcat .gsfc.nasa.gov with user guest. The DSP diskette images are available in directories DSPX where x is the number of the DSP diskette image. ENJOY and thanks for your support and continued interest.

REFERENCES

[1] McGwier, R., “DSP Modems”, *ARRL Sixth Networking Conference,* August 1987.

[2] Clark, T. and McGwier, R., “The AMSAT TAPR DSP Project”, *ARRL Sixth Networking Conference,* August 1987.

[3] Delanco Spry, Silver Spring, Md.

[4] Gardner, F. *Phaselocked Techniques,* Wiley, 1979.

[5] Clark, T, PACSAT project review, these proceedings.

[6] McGwier, R., and Price, H., PACSAT software, these proceedings.

[7] Johnson, L. and Green, C., PACSAT Computer, these proceedings.

[8] Poor, H.V., *An Introduction to Signal Detection and Estimation* , Springer-Verlag.

[9] TAPR, Inc., POB 22888, Tuscon, Az.,,USA

[10] AMSAT, Inc., 850 Sligo Ave., Suite 601, Silver Spring, Md. 20044, USA.

Digital Radio Networks and Spectrum Management

Paul A. Flaherty, N9FZX

Computer Systems Laboratory *and*
Space, Telecommunications, and Radioscience Laboratory
Department of Electrical Engineering, Stanford University
ERL 408A, Stanford, CA 94305

Abstract

Spectrum Management is a vital part of amateur radio. Questions of where to place services in the available spectrum continue to plague frequency coordinators. This paper contends that multiaccess radio systems should be allocated in the spectrum below one GigaHertz, and that monoaccess or link oriented systems be placed above that frequency.

Introduction

Electromagnetic Spectrum is a scarce, sometimes renewable resource. Much of the research in radioscience today is devoted to spectrum - efficient methods of communication, including such mechanisms as amplitude - compandored sideband telephony, and minimal shift keying data transmission. Only recently, however, has research touched on the area of spectrum reuse, and the impact of position within the radio spectrum considered.

Propagation characteristics of certain bands make those spectra valuable to classes of users. Ionospheric propagation below 30 MHz makes the High Frequency bands valuable to the world community. Small component size and portability are important to mobile users, and so the Very High and Ultra High bands play an important part in mobile communications.

Beyond these characteristics, however, little can be generalized about the appropriate spectra for certain classes of applicants. It is not readily apparent that one band should be preferred for multiaccess applications, and another for link - oriented systems.

Packet Radio is considered to be a spectrally efficient mechanism for digital communications. Using time - division techniques, several users may share spectrum without interference, if certain traffic characteristics hold, and if the network load is limited. Techniques for time - sharing spectrum abound, but all require some degree of omnidirectionality in the transmission or reception system, which is characteristic of all all mul tiaccess networks.

Using packet switching techniques, it is possible to construct a link - oriented, or monoaccess network, which is functionally equivalent to a multiaccess network. This duality can be exploited for networks with fixed or portable stations.

In a hierarchal networking architecture, the Terminal Network is usually defined as that hierarchy or subnet which connects to end users. The telephone local loop plant, and radio repeaters are two examples of terminal networks. This paper is primarily concerned with terminal networks, although many of the principles may apply elsewhere.

Synthesis

The forward gain of a parabolic reflector antenna is given as:

$$G=\eta\pi^2d^2\frac{f^2}{C^2} \qquad \{\text{Gain}\}$$

It is of no small consequence that the gain of a reasonably sized antenna increases dramatically with frequency; many digital satellite services exist explicitly because of this fact.

For the purposes of discussion, a "reasonably sized" antenna is considered to be unity, or one meter in diameter, for terrestrial applications. "Reasonable size" is often a matter of community tastes and economics; however, the one meter size covers a large portion of of the contingencies. Thus, the gain of reasonably sized antenna is:

$$G_0=\eta\pi^2\frac{f^2}{C^2} \qquad \{\text{Normal}\}$$

The half power beamwidth of a typical parabolic reflector is:

$$A=\frac{139}{\sqrt{G}} \qquad \{\text{Degrees}\}$$

Digital modulation schemes may be divided into two classes: orthogonal modulation techniques, such as phase shift keying, and antipodal modulation, such as amplitude or frequency shift keying. In order to add another bit per symbol in a constant - bandwidth channel, an increase in the signal - to - noise ratio of 3 db is required for orthogonal modulation, and 6 db for antipodal systems.

Frequency Division Tradeoff

The Frequency Division Tradeoff between multiaccess and monoaccess networks arises out of the increase in signal - to - noise ratio that occurs with the use of directional radiators. With the increase comes the ability to either multiply the bit rate, or divide the bandwidth to obtain equivalent service. Because antenna gain is tied integrally with frequency, the ability to fraction the bandwidth increases frequency, until a point is reached where each node occupies its own channel. The transition from a multiaccess network to its monoacccss dual occurs at a certain Critical Frequency, which is determined in turn by channel access technique, and network size.

As an example, consider a terminal network of eight. nodes, using a Carrier Sense - Multiple Access, and frequency shift keying, running at a rate of ***19.2*** Kbps. Assuming the best case for CSMA (no hidden nodes), the best aggregate throughput we can expect from such a network is about 10.6 Kbps.

The dual of this network is a set of eight links connected to a packet switch. Again assuming the best case for CSMA, each user has access to a 19.2 Kbps data rate. We wish to accomplish this transition using equivalent power and bandwidth; therefore, we require an eightfold increase in the aggregate bit rate. Assuming the use of n-ary frequency shift keying, this in turn requires an increase of 42 db in the signal - to - noise ratio. Such an increase can be obtained by a pair of one meter aperture antennas, operating at 1.5 GHz, using a 55% efficient feed. The aggregate throughput for this network is 153.6 Kbps, in the same bandwidth.

In general, for a large class of terminal networks, the Critical Frequency lies around one GigaHertz. The extent of the tradeoff is limited in practice by packet switching speeds, and the extensibility of multilevel modulation schemes.

Space Division Tradeoff

The propagation characteristics of radio limit the spatial dimensions of any network. However, it is often the case that the network itself covers far less territory than the radio spectra used to service it. This is particularly true with multiaccess networks which require omnidirectional radiators.

Radio propagation models are somewhat involved; the more exacting models have been implemented as computer simulations by researchers. However, even a cursory analysis reveals that spectrum reuse is much more practical at higher frequencies. In particular, path loss increases as the square of the frequency, as does antenna gain (which results from a narrower beamwidth). Wave polarity separation also increases accordingly. In general; it should be possible to model the multiaccess - monoaccess tradeoff, using the available computer tools.

As an example, consider the CSMA network mentioned earlier. The farthest node is at a distance R from the hub. In order to preclude the "hidden station" problem, stations on the circle described by R must have enough power for range 2R. In the limit, as the number of stations grows, the area covered by the radio network becomes four times as large as the area of the physical network. The monoaccess dual is no larger than physical network area at some Critical Frequency, and can indeed be: considerably smaller.

Towards a Spectrum Efficiency Quotient

Clearly, a combination of three separation techniques (spatial, spectral, and polar) can yield a spectrally efficient monoaccess network at higher frequencies. At lower frequencies, however, the multiaccess model predominates.

The term "spectrally efficient" has been used to describe multiaccess networks, without specificity. What is needed is a "figure of merit" to describe a radio network, and compare it with other alternatives. Propagation characteristics of the spectrum below one GigaHertz lend themselves to applications requiring a high degree of mobility and portability. For fixed or semiportable operation, however, a monoaccess network provides a spectrally efficient alternative, when operated above the Critical Frequency.

Summary

The spectral efficiency of monoaccess and multiaccess networks varies with the frequency used. The exact calculation of the Critical Frequency of the tradeoff is currently the subject of research. However, in general, multiaccess networks tend to be more spectrally efficient below one GigaHertz, and monoaccess networks predominate above.

Implications for the Amateur Service

Coordination between different types of services in the Amateur Service at frequencies above 30 MHz has been accomplished fairly haphazardly and ad hoc. With the advent of packet radio, it has been difficult in major metropolitan areas to coordinate use of spectrum. Repeater links have been traditionally placed in bands close to repeaters, because of the availability of equipment, and economy.

Ultimately, some changes need to be made in bandplans for the Amateur Service. In particular, it is recommended that stations in Auxiliary Service (as defined in Part 97.86) should be relocated to frequencies above one GigaHertz. Terrestrial digital links, used to interconnect multiaccess networks, should also be placed in the microwave region. In turn, multiaccess digital networks should be placed in the Amateur VHF and UHF allocations.

References

Wozencraft and Jacobs, ***Principles of Communications Engineering***, 1965, John Wiley and Sons, New York. ISBN 0-471-96240-6

William Stallings, ***Data And Computer Communications***, 1985, Macmillan Publishing, New York. ISBN 0-02-415440-7

Where is My High-Speed RF Part II

Terry Fox, WB4JFI
Vice President, AMRAD
7877 Hampton Village Pass
Annandale, VA 22003

Background

In 1986 this author presented a paper at the 5th ARRL Computer Networking Conference in Orlando, Florida regarding high-speed RF (or more accurately the lack of it.) That paper was meant to spurn on experimentation in the RF arena of packet radio.

The paper started with listings of several frequency coordinating bodies recommendations for packet radio operation. It then delved into the various channel access methods available, followed by a discussion of user versus backbone channel operation.

The last section described what was being used at that time, along with a few experiments being conducted in the high-speed arena.

This paper is meant to continue where that one left off. It seems we have made some progress, but have also stood still. While the last statement at first appears contradictory, it does describe what has happened.

What's Happened Since 1986

After the 1986 Conference, several things have happened. The first to be discussed will be the hardware and technical achievements, followed by the political and economic conditions.

Hardware In 1986 Paper

The 1986 paper described a few experiments that were known at that time. Only one of those actually reached any level of operation. That was the Steve Goode, K9NG modem. Steve had described in a paper given at the Fourth ARRL Mamteur Radio Computer Networking Conference how to modify a Hamtronics FM-5 220 MHz radio for 9600 bps operation. It turned out that this modification required a spectrum analyzer to accomplish properly, and the Hamtronics radio tended to have problems with wide temperature variations (such as found on television towers). The most reliable method of runnung the K9NG modems was to reduce the data rate to 4800 bps.

The attempt by AMRAD to design high speed equipment for 220 MHz met an early demise, based primarily on a combination of the lack of access to test equipment plus lack of coordinated time between the people on the project. This is a common occurance in Amateur Radio, compounded within AMRAD by the number of interests key players are involved with.

The Gary Fields equipment mentioned in the 1986 paper also never materialized to my knowledge.

The last technical item mentioned in the 1986 paper wasn't equipment, but rather a specification dreamed up by the ARRL Digital Committee. It specified a device that was felt by that committee as necessary to expand to higher speeds on various bands. The specification called for 56 kbps operation, with an in/out IF frequency of 28 MHz, which could then be transverted to the band of choice. More on this shortly.

Progress Since 1986

Since the 1986 Conference, some progress has been made. Some has also been promised.

Kantronics 2400 bps

Over a year ago, Kantronics introduced their KPC-2400, which is a TNC with a modem designed to run at up to 2400 bps, or twice the rate most Amateurs use. While it has been met with a hugh collective yawn by the Amateurs, it seems to be selling in the commercial arena. Most Amateurs realize that the! Kantronics twice-the-speed-of-sound arguement doesn't quite ring true, especially when channel turn-around times, among others, are taken into consideration. An important point to be made is that they have been shipping KPC-2400 TNCs for over a year, it is for real.

AEA 9600 bps 220 MHz Radio

Almost since the 1986 Conference there have been rumors that AEA was coming out with a combination radio that ran both voice and 9600 bps packet on 220 MHz. It was talked about at the 1987 and 1988 Dayton Hamfests, and has appeared in their catalog as recently as this summer. The radio has not shown up in dealer showrooms this author watches, but will supposedly Real-Soon-Now, or RSN, as a famous author likes to say.

GLB 19,200 bps 220 MHz

Almost since the 1986 Conference there have also been rumors that GLB Electronics is coming out with a 220 MHz radio that is designed for packet operation at 19,200 bps. Orders have been in place for over a year and then cancelled. Once again, RSN.

Dale Heatherington 56kbps IF Modem

Last year, Dale Heatherington, WA4DSY, came up with an IF modem that operates suprisin l like the abstract device described by the AWRE Digital Committee. In his paper he describes the modem design and operation including scope pictures. It is a Minimum-Shift-Keyed (MSK) modem that uses a roughly 70 khz bandwidth channel. It puts out about 1 mw at 28-30 MHz. It contains both the modulator and demodulator.

The best part of this modem is that boards (and now kits) are available to build the units. It is not cheap to build one, the parts cost about $250. Then the unit must be assembled and aligned, which can also be tricky without proper instrumentation.

While Dale describes one design goal as having no exotic parts, there 'have been some problems obtaining a few of them.

Also, keep in mind that even after the modem is built and aligned, the output is still at 28 MHz. a transvertermust also be purchased to move the modem output to the frequency of choice.

While the above indicates the relatively complexity of obtaining high speed operation, it must be said that Dale amd his associates have come through with a real benefit to packet radio. Keep in mind these modems are not for everyone, but may help greatly in speeding up the inter-bulletin board traffic on backbone channels.

By the way, as Dale indicates, one of the toughest problems related his to high speed modems is the generation of real packets at that speed. Suddenly the shift is back to the digital people!!

Use of 900 MHz

Recently, an idea came to this **author at** one of the weekly AMRAD Taco meetings. **By the** way, a couple of years ago AMRAD **decided** to forego the standard pizza meetings in favor of something more spicy. **It** is **unclear** if this has helped **to** induce our present hopping from one project to another, but the suggestion has been made.

With the present Amateur band situation, it has come up once again that the 900 MHz Amateur band is almost devoid of use. **Other than a** few experimenters, not much is **happening up there** at least in the Washington D.C. area. The reason' for this is simple, no equipment is available off the shelf. Reality is that Amateurs today buy almost all of the RF equipment they use. If we are to rely on homemade equipment to populate a band, we will surely lose that band. There is over 20 MHz at **900** MHz just waiting for users, be they Amateurs or others.

Now for the idea. **THe** idea actually was formed in two parts. The first was reading a Sunday paper, and the second was at a Taco meeting. The explosion of cellular and **trunking** radio **systems has** brought with it inexpensive **RF** equipment operating at 900 MHz. Up to 5 watts is generated in cellular telephones, cheaply. In **several** recent Sunday papers there-were advertisements for new cellular telephones for as cheap as **$399!** Most Amateurs pay more than that for a decent HT and batteries! Imagine taking these **cheap** cellular telephones and gutting most of the digital electronics, then **using the** RF section on the 900 MHz Amateur band, with packet data sent instead of voice.

Or possibly a transverter that mixes the output of a Dale Heatherington 56 kbps modem with the RF from a low-level stage of a cellular telephone, and then amplifies the result with the rest of the telephone Rf circuitry (it should be broad enough). The opposite procedure might be usable in receive.

By the way, there maybe a market for these cellular phones in non-packet Amateur Radio. If someone could build an Amateur repeater or group of repeaters that takes advantage of the cellular ditigal and RF technology directly...

High Speed Voice-Grade Audio Modems

Another alternative for high speed packet is with the use of high speed audio modems, such as those presently being used over tele hone lines. If thesemodems really send data as 4ast as some claim over regular dial-up telephone circuits, the possibility exists for their use in Amateur Radio. There are some problems with this, but they may not be insurmountable.

The first and immediate problem is that of distortions created in the radio portion of the connection. Most Amateur rigs are not designed taking phase errors into account. Quite a few don't even perform well in frequency response. Most of the higher speed modems use some sort of phase, amplitude, or frequency modulation, or a combination of those. Any radio used with these modems must be capable of accurately reproducing the modem signals at least as well as a **telco** circuit.

Another problem with the use of these modems is that they **typically** needs **some amount** of time to lock to **each other.** Since phasing between data chunks is often used,' not only must the frequency be correct, but a reference phase must also be generated. In full-duplex telephone lines, once the connection has been made at the beginning of the call and the modems have achieved lock, there is a constant reference to maintain lock. In the Amateur world of half-duplex, the modem signal is not always present. In addition, if a third station was to join the first two, it will confuse the other two, requiring a re-lock (sometimes referred to as the training period) each time a different station transmits. These training periods are based on the modem technology, and may not be directly related to data rate. Sometimes the training period can be in the order of many seconds. In that case, even though the data rate is quite fast, the training period is sufficiently long to offset it.

Given the above discussion, these audio modems do not appear to buy a lot of long-term speed. Considering they are also costly, they may not be a real alternative at this time. They should be monitored however for breaks in technology, or if conditions favor them (such as a direct path between only two stations with decent radios).

Politics and Frequencies

The reality is that we Amateurs enjoy a large amount of spectrum space, placed up and down the frequency spectrum. We also have had a great influence on the world of communications, due in large part to our experimentation within these bands. We HAVE **paid** the dues to justify the spectrum.

Some do not share this opinion. Others see a method of making money **using** *additional spectrum. Our contribution is not **directly** related to monetary gain (per FCC **regulations),** so **in todays** world we are often the odd man out. **Do not** believe for one minute the loss of a portion of the 220 MHz band is the end of raids upon us. Quite the opposite. If it is upheld, more may go. Precedents are dangerous things.

220 MHz Band

When the 1986 paperwas written, almost all **packet** activity was limited to the two meter band. Very few Amateurs had expanded beyond those frequencies. Since then there has been an explosion in the number of packet users, and the number of packet bulletin boards. Many of these bulletin boards have shifted to the 220 MHz band for message handling between themselves.

The recent action taken by the Federal Communications Commission that removed the lower 2 MHz of the 220 MHz band **from Amateur** service has thrown much of the orderly progression of packet radio right out the window. Up to the point that the FCC made this decision, there was real hope that the Amateur community might finally see some faster packet links. Now, the implementation of these **high** speed channels is in doubt.

This author is among the may Amateurs that now own useless 220 MHz radios **if** the FCC action stands, in my case three radios. Some Amateurs **suggest we** not give up on 220 MHz. We might get **it back. My opinion is** that we must start planning **for the** future. If we get to keep 220 MHz after all, so much the better. My **220** rigs aren't going anywhere. They can't. Nobodywants them. Oh well. And we just got in the right crystals too.

420-450 MHz Band

Due to the FCC action regarding **220** MHz, the 450 MHz Amateur band has suddenly **resurfaced as a** place to put high speed **packet radio.** Unfortunately, this band is already overcrowded **in** most metropolitan areas, especially with ATV onerations. It **appears** this band has too much activity **already.** Even so, some Amateurs are planning to move the existing 220 MHz links to **450** MHz if the FCC action is not reversed.

There are also persistent rumors regarding the loss of part of this band. In addition, the Canadian border has zones nearby where operation on 450 is restricted. It is this authors opinion that long term operation on 450 MHz is unlikely.

900 MHz Band

The next Amateur band is 900 MHz. As previously stated, the foremost problem here is the lack of equipment. In addition, path loss is greater. This can be offset somewhat with higher **gain** antennas if directionality in not a problem. Given the present state of politics, the 900 MHz band **would** most likely be the best spot for high speed packet operation if the equipment hurdle can be overcome. Another possibility for this band when equipment becomes **available** is to **actually** move the individual Amateurs to 900 **MHz, allowing** better cellular-type RF operation. Presently this is not possible, not only due to the lack of equipment, but also because of the lack of network resources to interconnect users.

1.2 GHz and Above

Frequencies above the 900 MHz band start getting scary to most Amateurs. Not only is the equipment difficult to understand and operate, test equipment becomes hard to locate and employ. Some Amateurs do not have these problems, but the number of these types are far too few to rely upon for building and maintaining packet network resources. Eventually, as speeds increase, we may need to use these higher frequencies for point-to-point microwave relays much as telco has for years. That time has not come yet, however. (Here's hoping someone proves me wrong!!)

Conclusion

While packet radio has been growing in numbers, not enough work has been done in the RF arena. In 1986, I was hoping we would be further along than we are. Because of the recent FCC action, we might even go backwards. The biggest single contribution made since my 1986 paper has to be the Dale Heatherington modem. It is almost exactly what the ARRL Digital Committee spelled out. It works, is reproducible, and can now be purchased in kit form. It is not limited to a single Amateur band. Some companion RF transverters would be a very welcome contribution.

Much more work still needs to be done. Several efforts have been started but not completed. The two commercial digital radios (AEA and GLB) may have to be re-engineered for a different band (thanks FCC!).

As I ended the 1986 paper, so too I end this one. If this sounds like I am begging, I am. I WANT FASTER RADIOS!!

References

Fox, T.	"RF, RF, Where is My High Speed RF", Fifth ARRL Amateur Radio Computer Networking Conference, ARRL, 1986
Goode, S.	"Modifying the Hamtronics FM-5 for 9600bps Packet Operation", Fourth ARRL Amateur Radio Computer Networking Conference, ARRL, 1985
Heatherington, D.	"A 56 Kilobaud RF Modem", Sixth ARRL Amateur Radio Computer Networking Conference, ARRL, 1987

Proposed AX.25 Level 2 Version 2.0 Changes

Terry Fox, WB4JFI
Vice-President, AMRAD
7877 Hampton Village Pass
Annandale, VA 22003

Purpose

This paper is presented to document some of the major changes proposed over the last four years of operation for the AX.25 Level 2 Version 2 Protocol. These changes have been collected by this author from various sources, and were recommended by a working group of the ARRL Digital Committee which met in July of 1988.

Background

The Amateur Radio Link Layer packet standard known as AX.25 Level 2 has come along way since it's creation during the summer of 1982. At that time, six Amateurs
discuss a replacement for the original Vancouver Protocol created by Douglas Lockhart, VE7APU. Since this author had already written several of the proposed ideas in the AMRAD Newsletter, a draft of the new AX.25 Level 2 was begun.

This draft was almost completed when Tom Clark, then President of AMSAT, called a meeting in October of 1982 to establish a standard Link Laver Protocol to be used on AMSAT Satellites. At that meeting, several protocols were discussed, with the result being that a slightly modified version of the AX.25 Level 2 Draft-being adopted. The major alteration was to include fields for more than one digipeater. This draft was then used by TNC-1 Terminal Node Controllers. The AX.25 draft quickly became THE standard for packet radio due to it's implementation by TAPR. It should be noted that the TNC-1 was capable of both AX.25 and the original Vancouver Protocol. The AX.25 Level 2 Protocol Specification was published in March of 1983 in the Second ARRL Amateur Radio Computer Networking Conference Proceedings by this author.

The AX.25 Level 2 protocol was heavily based on the commercial X.25 Protocol Specification, with some revisions. One of these revisions was the removal of certain types of frames (S-Frames) as commands for link status and maintenance. Instead, Information Frames were used, along with a heavier emphasis on timers. This was done to simplify the protocol implementation, but as it turned out, this short-cut caused more problems than it relieved.

It became apparent that some of the changes, such as the removal of command S-Frames, were not working as well as had been expected. About the same time the ARRL and it's Digital Committee expressed a interest to officially adopt the AX.25 Level 2 protocol. These factors led to the creation of a new version of the protocol, which more closely adhered to both the CCITT X.25 and AT&T BX.25 standards. The original specification was expanded and refined to become AX.25 Level 2 Version 2.0, which was adopted by the ARRL Digital Committee and the ARRL Board in October of 1984. The original AX.25 was labeled Version 1.0 to indicate it's operational differences.

Version 2.0 of AX.25 Level 2 has been in use for four years now, with estimates of between 50 and 80 thousand devices using it. During this time, some additional problems with it have been encountered. The remainder of this paper discusses alterations to the AX.25 Level 2 Version 2.0 Protocol Specification.

Backward Compatibility Issue

As just mentioned, there are an estimated 50 to 80 THOUSAND devices using the AX.25 Level 2 Version 2.0 protocol. A fundamental issue that must be resolved is what happens to these users when alterations to the protocol are made. In Amateur Radio, there is no method of requiring this installed user base to migrate to a new protocol quickly. If changes are made that cause incompatibility between users, complaints are sure to follow. An example of this is found with the TAPR TNC-1. Due to an oversight in the software design, TNC-1s cannot be used with Version 2.0, even as digipeaters. This is because the TNC-1 code accidentally tests bits that were clearly marked as reserved in the protocol spec. which were later defined in Version 2.0. To this day, people complain about this, even though it is an implementation problem, NOT a protocol issue.

Some of the more agressive protocol changers do not believe that backward compatibility to previous versions should be an issue. They feel changes for the. better should be made automatically without regard to on-air compatibility. In fact, they argue, if the newer version causes enough on-air problems and crashes, the rest of the community must follow just to stay on the air. This author vigorously disagrees with that method of subterfuge. Another potential result of this scheme is the turning-of fofboth network resources and Amateurs themselves.

Due to the large number of devices using Version 2.0, this author feels that backwards-compatible solutions should be implemented whenever possible. This guideline should be first and foremost. If a solution cannot be reached which is backward-compatible, it should be flagged and carefully weighed before implementation, since that implementation would immediately cause at a minimum incompatibility, and possibly complete and systematic crashes.

It should be noted that the above relates primarily to communications channels where large numbers of Version 2.0 users are found, and is NOT meant to preclude either experimentation or modifications to enhance specialized channels, such as high-speed network backbones. Since there will be a limited number of users on such a channel, they should be allowed to run any agreed-upon changes or enhancements which are within the governing body's rules and regulations.

In the following discussions, two distinct types of modifications will be discussed. Those that would be backwards-compatible to Version 2.0 could be considered for either a Version 2.1 release or a Version 3.0 release, while modifications that would NOT be backwards-compatible should be considered for a Version 3.0 release ONLY. This numbering scheme follows this author's recommendation that minor revisions are noted by the number after the period, while major alterations are noted by the number preceeding the period.

Addressing Issues

One of the prime motivating forces behind the move to update AX.25 Level 2 is its limitations in the area of addressing. When this author originally proposed the use of an extended address field that included Amateur callsigns, it was felt that six characters would be of sufficient length to contain the callsign, with a seventh tacked on to allow each Amateur to maintain more than one station. It now appears that both prefixes and suffixes should be sent in addition to the callsign. These additions often cannot fit within the six octets allowed under Version 2.0.

Further, since Version 2.0 specifically calls for sub-fields of six characters plus SSID, extending the individual callsign address sub-fields is precluded.

After much individual research and the suggestions of many others, two methods of allowing adding the additional information have been recommended. The first method has been designed to be backward-compatible, but is rather inelegant. The second method is designed to cure regulatory-related problems outside the United States in addition to the above issue, but is NOT backward-compatible.

Backward-Compatible Addressing Extension

The first method of adding more addressing information to AX.25 Level 2 Version 2.0 is admittedly very much a kludge. It will not be immediately obvious to older equipment exactly what is going on, but it will function properly with the older versions, assuming they followed the previous protocol standard, unlike the TNC-1 as described above. This unwitting inter-operation with the older standard makes it backward-compatible with them.

Simply put, method one defines additional address sub-fields called extension sub-fields which, if present, convey the additional addressing information required. They are placed after the destination and source addresses, but before any repeater address sub-fields. As with the other address sub-fields, these extension sub-fields must be seven bytes long. Both the placement and encoding of the SSID in the extension sub-fields are a subterfuge to imply to an older version device that the extensions are digital repeater addresses, allowing the older version to ignore the extensions presence. Since Version 2.0 allows at most eight digital repeaters, any extension sub-fields must be subtracted from the number of allowed digital repeaters to keep the maximum number of "repeater" sub-fields at eight or less.

Address Extension Information Encoding

The additional information is encoded in the same manner as the other address information. It should be bit-shifted ASCII, stuffed with trailing ASCII spaces as required to six characters plus SSID. The only other requirement is that if the additional information is a prefix to a callsign, the slash (/) character is placed after the prefix. If the additional information is at the end of the callsign, the slash is placed before the postfix. For example:

Amateur Callsign	Normal Address	Extension Field 1	Extension Field 2
WB4JFI/KT-1	WB4JFI-1	/KT	(none)
VP2M/WB4JFI-1	WB4JFI-1	VP2M/	(none)
VP2M/WB4JFI/KT-1	WB4JFI-1	VP2M/	/KT

As implied above, more than one extension may be required, and may be used. If both pre- and postfixes are required and are under six bytes in total length (included a shared slash), they may be combined in a single extension field.'

The SSID number of the first address extension sub-field for each address (callsign) will be set to zero, the second to one, and so on as necessary. Not only will this aid in "glueing" the address back together, but will also indicate when one extension block ends and another begins.

The equivalent of the H-bit (bit 7) in the SSID octet of all extension sub-fields should be set to one at all times by a Version 2.1 device, indicating to a previous version device that this is a repeater field that has been repeated. This will allow a previous version device which is a destination to conclude that all repeaters (including the extensions) have repeated the received frame, and it may process the frame.

Extension Information Indication

Any address that has extension information will indicate this by resetting bit 6 (hereafter called the A-bit, for Address extension) of its SSID octet to zero (0). This bit has previously been reserved and should have been set to one (1) as indicated by both Versions 1.0 and 2.0 of the AX.25 protocol specification.

The extension su'b-address fields should also have the A-bit set to zero to simplify the comparison of extension sub-fields with repeater sub-fields in Version 2.1 devices.

Extension Information Placement

In order to fake-out earlier versions of the protocol, the extension information cannot simply follow its base address. The only place this new information can be placed which will work with older versions is between the source address and the first of any repeater addresses. If placed there with the H-bit set to one, older versions will assume the field is for a repeater that has already repeated the frame.

The order of appearance of these extension sub-fields is the same as the main address sub-fields; any destination extensions come first, followed by any source extensions, then any repeater extensions in the same order as the repeater sub-fields themselves.

Examples of Address Extension

The following example will aid in indicating the proper operation of the proposed address extension recommendation.

The following example indicates how a single address sub-field may be extended. In it, the destination field has a post-fix modifier. The frame is a UI command frame from WB4JFI-1 to K8MMO/KT-0. Note how the A-bit is set to zero both in the destination sub-field AND the address extension sub-field. Note also how the address extension sub-field has an SSID of 0000 indicating it is the first (and in this case only) extension sub-field.

```
! DA      !  SA      !  Ext. 1 ! CTL !
! K8MMO 0 ! WB4JFI1 ! /KT    0 ! UI  !
```

The actual bit-encoding of these fields is as follows:

	Al	K	1001 0110	96
D	A2	8	0111 0000	70
e	A3	M	1001 1010	9A
s	A 4	M	1001 1010	9A
t	A5	0	1001 1110	9E
'	A6	SP	0100 0000	40
	A7	SSID	1010 0000	EO
	--	--	CARS SID-	--
S	A8	W	1010 1110	AE
o	A9	B	1000 0100	84
u	A10	4	0110 1000	68
r	All	J	1001 0100	94
c	Al2	F	1000 1100	8C
e	A13	I	1001 0010	92
	A14	SSID	0110 0010	E2
-	--	--	CARS SID-	--
E	A15	/	0101 1110	5E
x	A16	K	1001 0110	96
t	A17	T	1010 1000	A8
e	A18	SP	0100 0000	40
n	A19	SP	0100 0000	40
'	A20	SP	0100 0000	40
	A21	SSID	1010 0001	Al
	--	--	---- ----	--
	CTL	UI	0000 0011	03

If there are real repeater fields in addition to the extension information, the address field will look as follows:

Uplink to the Destination Station

```
!  DA     !  SA     !  Ext. 1 !  RPTR 1 ! RPTR 2!
! K8MMO 0 ! WB4JFI1 ! /KT   0 ! WB3KDU5 ! WB4JFI5!
```

Link Back to the Original Station

```
!  DA     !  SA     ! RF'TR 2 !  RPTR 1 ! Ext. 1 !
! WB4JFI1 ! K8MMO 0 ! WB4JFI5 ! WB3KDU5 ! /KT   0!
```

If the link-back frame is passed through older version digipeaters, the H-bit in the Extension sub-field and A-bits in both fields may not be set to indicate an extension is present. Fortunately, since the newer device is now **receiving** the frame it knows there may be extension information which may look like repeater sub-fields. It can **compare** the frame's repeater addresses (or count the number of H-bits set compared to repeater fields used), to find out if all repeaters addresses have been used.

Address **Extension** Operation

If an older-version device starts the link, it will not create these extension **sub-**fields. Therefore, frames it generates, and those frames that are passed through repeaters will not have problems. If a new-version device is at the destination, it may respond with a **frame** that does have extension information. **The** extension sub-field will have the H-bit set, so the first station (and any repeaters) will **assume** it is just a has-been-repeated field. **If the** first station does not modify its path table entry, both stations will operate as first started, with the extension information being carried as "excess baggage". If the first station does **modify** its Path table to include the extension information as repeaters, the extensions will be placed at the end of the address field, allowing-any actual repeaters to repeat the frame properly. Once the frame has finished being repeated, just the extension information is left with the **H**-bit not set. The second station can evaluate the received frame and detect that all repeaters are done.

If the first station originating the connection is a newer protocol device attempting a connection to an older protocol device, operation is similar to the above. The extension information is placed between the Source and any repeater address sub-fields by the originator. Since the H-bit has been set on the extensions, if a repeater is involved, it will skip the extensions to the first actual repeater address without the H-bit set and test for its address. It will repeat the frame if its address is found, **setting** the H-bit when sending the frame out. This will continue until all repeater fields have been used, and the frame arrives at the destination (hopefully). The older device will reverse the order of repeater sub-fields. and send an acknowledgement **back.** The state **of the** A-bit in all fields is unreliable at this point, making it useless for further **processing. In** addition, the H-bit will be off on all repeater AND extension sub-fields. Since the extension **sub-**fields are now at the end of the address field. any true repeaters in the path will repeat the frame (and set the corresponding H-bit) properly. Once the frame has cleared all-the repeaters and arrived back at the first station, there are still repeaterlike **sub-fields** with the H-bit NOT set. **The new** version device must look at these to-see **if they** are true repeater addresses or **just** extension information.

New Address Framing Technique

The second method is the **subject of a** separate paper found elsewhere **in these proceedings** and will not be discussed "in depth here. It involves separating the addressing issues from the rest of the protocol, and totally redefining the address portion of Level 2 frames.

One of it's advantages is that the Amateur address portion of the frame is totally separate from the X.25 protocol machine, with counters and pointers relating to the address information maintained within the address portion of the frame, rather than implied, which is the **case** with AX.25 Version 2.0. This is meant to simplify software implementation of the protocol.

Another advantage is the addition of a hashed version of the next destination address as the first byte of the address field. This allows the implementation of selective addressing which is built into most synchronous hardware chips, reducing processor overhead.

In addition, Address sub-fields may be of variable length, and may include address mnemonics in addition to Amateur callsigns.

The main disadvantage of this new addressing **technique** is that **it** totally alters the addressing of **AX.25,** rendering it **absolutely** incompatible with ALL earlier versions. **This** removes it from the backwards-compatible **category.** In fact, since the alterations are so drastic, it may cause catastrophic failures of earlier versions just by being heard on a channel, especially to systems whose software is not designed to be fault-tolerant.

Another disadvantage of this new address scheme is that the HDLC address extension bit is NOT implemented, which means it technically violates the HDLC framing standards. The end of the address section is indicated by counters and pointers rather than by the use of the E-bit. The non-use of the E-bit means, among other things, that Protocol Analyzers can no longer be used to troubleshoot and analyze Links and software implementations.

The use of counters and pointers to indicate important places in the address field is meant to simplify software processing of the frame. This simplification is gained at the cost of additional channel overhead. Each of these counters and pointers which must be transmitted ties up the RF channel for that much longer. It is a small point, but does add up eventually.

In addition, there are other **pieces** of information which add to channel overhead, such as the hashed next-destination address field, an address checksum and more **protocol identifiers.** The total of additional information required is **a** minimum of 10 bytes above the AX.25 Version 2.0. Keep in mind that this is 10 bytes more in each transmitted frame.

It is obvious that in the above proposal, concern for **processing** power is higher than concern over channel overhead. Both of these are important issues, so any trade-off between them should be judged **carefully.** Generally speaking, as the channel speed increases, small **additional** overhead becomes less important while processing speed becomes more important. The inverse is true as channel speed is **decreased.**

While both of these address extension systems will convey the information required by the regulating bodies, the second method is not only more flexible, it is also easier to understand and implement. Is this trade-off worth the small additional channel overhead? Are the advantages also worth obsoleting all older versions of the protocol, at least on the channel it will be used on? Will both versions be required to be available simultaneously? Time will tell.

State Description Logic **(SDL)** Diagrams

In the back of the AX.25 Level 2 Version 2.0 Document are three state tables, which are meant to describe the operation of the protocol based on various external actions taking place. There has been some concern as to whether these tables are actually part of the document or just an implementation guide. They were included to easily indicate to implementers how the protocol should operate, and therefore are part of the protocol description.

One of the problems with these state tables are that they cannot easil indicate the various steps taken when an externaI event happens. There is only enough room to indicate if a frame is to be sent as a result of the event, and any state transition made as a result of that event. If there is more to be done, the "flatness" of the tables precludes **description** there, requiring the document reader to search the actual text.

There has been another method of describing the actions taken by protocols that is gaining in popularity. This method is called State **Description** Logic, or SDL diagrams. Most of the newer protocol documents developed by the **CCITT** are using SDL diagrams to describe protocols. **An** SDL diagram looks much like a software program flowchart, with slightly different symbols. These SDL diagrams are MUCH easier to read, follow, and implement from. There is an effort to document the AX.25 Level 2 Version 2.0/2.1 in SDL diagrams. If they are available in time, the next printing of the AX.25 Level 2 Document may include the SDL diagrams in place of the State Tables. The main reason for any delay is that the SDL diagrams are quite a bit more complicated, and must be checked very thoroughly with the text to make sure they agree with each other.

State Table Changes

Given the preceeding information, State Table changes may be a moot point. They are included here for completeness.

The main State Table change is the removal of States 14 and 16. After review, no one has ever found how a protocol machine could remain in either of these states for any length of time. Both have to do with the local station being busy, but having sent a REJ frame. The sending of the REJ frame itself indicates a non-busy condition by requesting a re-transmission.

Unumbered Information Operation

There has been some discussion regarding the proper use of the Unnumbered Information, or UI frame. This is especially true when discussing the use of the Poll bit associated with UI frames. Some feel that it is possible to maintain a separate "UI Mini-State-Machine" by the use of Poll and Final bits associated ONLY with UI frames. After careful review, it was deemed that in order for these P/F bits not to interfere with the normal protocol machine P/F operation, more processing overhead would be required than could be justified. Therefore, UI frames are left as Commands, with no Poll bits used.

Automatic Re-connect Elimination

Perhaps the largest complaint heard regarding AX.25 Level 2 performance is when a "disconnected" connection keeps coming back. THere are cases when a station has requested a disconnect, gone into timer recovery due to the disc frame getting lost, and then receives I frames from the other station. Eventually the first station will consider itself disconnected, and send DM frames. The second station still has data pending, so it will re-establish the connection and pass the data. This is NOT a bug, rather it is a deliberate attempt to make AX.25 re-establish failed links. For our shared RF medium, it now appears this tactic is too aggressive. The new recommendation will specify that the link is not re-established, instead an error message is passed to the higher layer protocol or program, which will decide what action the AX.25 Level 2 machine should take.

Maximum Packet Size

There have been many discussions regarding the maximum allowable frame size. Some people feel the 256 byte limit of data in Information frames is too small. They site two reasons for an increase in size. The first is that some Higher-Layer protocols require substantial overhead (such as TCP/IP), and this overhead must be added to the real user data, subtracting from the total transmittable user data per frame. The other reason is that higher speed channels should be able to transfer larger amounts of data per frame, increasing the overhead-to-user-data ratio, thereby making the channel more efficient. Information field sizes of 1024, 2048, or larger have been suggested.

Making an ad-hoc change to the maximum frame size may have serious repercussions, however. Older devices amy have hard limits to the maximum allowable frame size, reducing the chance of backward compatibility. An even greater potential problem is if implementations do not check for an allowable maximum frame size and crash if substantially large frames are received. Older devices have a limited amount of RAM memory, and substantially larger frames might also tie up too much memory, another potential for crashing implementations.

In order to make sure older eqipment doesn't crash due to excessive frame sizes, there needs to be more study done on this issue. Otherwise, drastic results may occur.

The result of discussions within the Digital Committee is an agreement that the 256 byte limit be maintained at this time, with an escape clause that Source, Destination, and Intermediate repeaters may use agreed-upon larger frame sizes on particular link connections and channels. Implementers should be aware of this, and make sure maximum allowable frame size is checked.

Maximum Window Size of One

Phil Karn has been suggesting that present AX.25 connections that send more than a single frame per RF transmission are actually using the channel inefficiently. He feels there should be only one I frame outstanding at a time per direction per link, creating his Ack-Ack protocol. Even if there is some accuracy in his arguement, altering the protocol to make this operation mandatory would be short-sighted. The decision was to leave it alone at this time.

Non-Use of Polling

The original AX.25 Level 2 Version 1 specification did not use Supervisory frames as commands, only responses. Their introduction as commands was the significant alteration that caused Version 2.0. Phil Karn now suggests that we go back to the original system, using I frames for retransmission recovery, but only if the lost I frame was "small". After some discussion regarding "small" vs "large", and implementation requirements, it was decided to keep the Version 2.0 scheme for now. It should also be noted that this system follows the traditional X.25 approach.

Forced Disconnect

Franklin Antonio has raised an issue regarding the Disconnect Request state, S4. Presently, while in this state, if a Local Stop Command is received (from the higher-layer), no action is taken. He suggests a transition to the Disconnected state, S1, and the discontinuation of sending Disc frames. This appears to be reasonable behavior, and was recommended.

Stop Timers During Channel Activity

The main reason for the use of timers during AX.25 Level 2 links is to make sure the link is still valid during slack transmission periods and for error recovery. In half-duplex Amateur Radio use, errors are normally introduced whenever two or more stations transmit at the same time, interfering with each other. The timers presently run whether the channel is busy or not. Some Amateurs argue that whenever the channel is busy, a station cannot transmit anyway, and the busy period should be removed from the delay period. This may also add to the randomization of the delays before station retransmissions.

Stopping the timers appears to have some advantage, but its implementation may cost more in processor overhead than is gained. This issue has been reserved for further study before a decision is made one way or the other.

Ack Prioritizing

Another suggestion that has been made is to make sure acknowledgements have a higher priority than other frames during connections. It is suggested that this will cause fewer retransmissions, since the shorter acks can be clobbered by longer data frames, causing error recovery procedures to be hastily implemented.

After reviewing the various requirements to implement this, especially when repeaters are involved, it was felt that implementing Ack Prioritization could become extremely complicated. More study is needed regarding this subject, as it may still have advantages.

Remote AX.25 Level 2 Parameter Control

There have been a few Amateurs that have indicated a need to be able to remotely access and possibly alter various Level 2 parameters. This has been met with quite a bit of resistance, primarily because of the potential for link damage by others, either accidentally or on purpose. If remote parameter setting is to be performed, it should be done within the confines of a higher protocol, preferrably with some authentication.

Implementation Issues

There have been several questions asked regarding how to properly implement AX.25 Level 2 Version 2. There have also been some "bugs" introduced on the air due to its improper implementation. After discussion of a few of these implementation problems, it was decided to include an Appendix to the AX.25 Level 2 document discussing implementation issues. In addition, the inclusion of the SDL diagrams discussed above should help resolve future questions. A few of the implementation problems are discussed below.

Queued Text When S1 or S4 Entered

Franklin Antonio points out that some implementations (particularly TNC-1 and TNC-2) send any queued text left upon disconnection as Information in UI frames AFTER the disconnection. This should NOT be done. Either the link should be re-established, or the information should be discarded (the decision on which action to take is now left to the upper-layer). This is an implementation error.

RNR and Memory Usage Problems

Some Amateurs indicate that there is a problem with some devices which use RAM memory to store, or log, received data or messages. After a certain amount of operation, these devices can become full of data. If a connection is established while one of these devices is already full, that device will allow the connection, but then indicate its lack of resources by sending RNR frames until the user frees some memory. This should not happen. If a device cannot support a connection, it should indicate that by rejecting the connection request (with a DM response). This is clearly an implementation issue that is beyond the protocol document to describe. The document is meant to describe how the protocol machine operates, not all possible implementation issues, such as mail storage in the same device as the protocol machine.

Bells, Clear Screens, Other Binary Data

A problem that is becoming more an issue every day is that of binary data being passed to the user terminal from the TNC device. Quite often, a user who is monitoring a packet channel suddenly hears bells, the screen clears, and other strange things start happening. This is often because more real network protocols are showing up on the air. These network protocols use binary data in their control fields (which are located in the information field of the AX.25 Level 2 frame), which can cause a terminal to go crazy if it receives the binary data.

When the AX.25 Level 2 protocol was designed, a Protocol IDentifier (PID), was added to indicate what type of higher level protocol if any was being used. At that time, a PID of FO hex was issued to be used whenever no upper layer protocol is being used. Since then, additional PIDs have been assigned to Network Layer protocols. At the time it was hoped that if a device (or software between the user and the device) saw a PID other than FO, that device could selectively not allow the data to the terminal (or computer screen). Since this was not clearly stated as a purpose in the protocol document, it appears that was not implemented. Future software designers should allow for this option, reducing the number of problems showing up on the user screen. This will become even more important as more network protocols show up on the air.

Level 1 and Level 1/2 Interface Issues

Many of the proposed changes received are not directly related to AX.25 Level 2 operations. Specifically, adjustment to L2 timers and channel access operations should be considered outside the scope of the AX.25 Level 2 protocol documentation, since different channel access protocols require different settings and adjustments. There needs to be more work done in this area, possibly as a separate AX.25 Level 1 document. The following is a list of ideas and suggestions that should be considered for a Level 1 document. As an alternative or interim solution, either an addendum to the AX.25 Level 2 document, or a separate working paper could be created to indicate the suggested operations. Another reason for this separation is that most of these adjustments are to fine-tune half-duplex CSMA or CSMA-CD operation. In full-duplex, slotting, or other channel access methods, these parameters do not necessarily come into play, and do not need to be specified or adjusted.

If there is separate document created for these Level 1 issues, wording should be added to the AX.25 Level 2 document indicating which parameters may be altered as a result of another Level's operation.

Persistence

Back in the old days of Vancouver boards running either V1 or AX.25, the software would not automatically and aggressively retry, but would wait before retransmitting. Unfortunately, the TNC-2 software did not implement this, and as a result there has been a beating-of-the-chest regarding schemes to take care of this "sudden" problem. Recently, Phil Karn and others have rediscovered this situation and has suggested we modify the Level 2 specification to include p-persistence.

Adding requirements to the AX.25 Level 2 specification regarding persistence would be a mistake. We will not ALWAYS be using half-duplex CSMA as the channel access protocol.

In our half-duplex environments, p-persistence should be used, with the value of the persistence being set depending on channel operation. The actual value and rate of any alterations made to the persistence are subject to further study.

Retransmission Backoff

Another Level 1 issue has to do with the adjustment of the retransmission timer, T1. Originally, the Vancouver TNC also added some delay each time a retransmission was attempted. Exponential backoff was insisted upon by Phil Karn last year, he has since agreed that exponential backoff may be too aggressive. Tom Moulton has indicated that some have found that on marginal links a simple arithmetic backoff may operate much more efficiently than an exponential backoff. Frnklin Antonio also states some reservation regarding exponential backoff, and mentions arithmetic backoff as an alternative. After some discussion regarding actual implementation of retransmission backoff, most agreed that second-order polynomial backoff might be a good compromise. This may still be subject to further experimentation.

T1 regarding Round-Trip Timing

Yet another modification to the T1 timer Phil has proposed is to have it's value adjusted based on an average of the round-trip time for information packets sent and acks received. Once again, the old Vancouver AX.25 code had a variation on this, in that it automatically altered the value of T1, based partially on the number of digipeaters used by the link. Phil suggests that a continual monitoring of round-trip timing be used, and a smoothed version of this value be used to adjust T1. Retries should not be included in adjusting T1, as that may throw off the actual round-trip timing.

Carrier Sensing

Franklin Antonio, N6NKF, has suggested that RF carrier sensing be used in half-duplex operation. He points out that while most all AX.25 software presently implements this for half-duplex operation, it is not actually written in any specification. This is yet another item that belongs in another Protocol document, related to channel access methods. This is especially true since carrier sensing really applies primarily to CSMA channels. Full-duplex, slotting, and Aloha may not need to do this sensing.

Keyup Delay

Most TNCs have a variable delay between when the RF transmitter is first turned on, and when data actually starts being sent, to allow for the transmitter to stabilize. This keyup delay is often called TXDELAY, and Franklin suggests that it be required to be user-settable, and he also suggests that if possible it be implemented (at least partially) in hardware, due to the wide variation of values based on packet speed. While its specification is needed, Keyup Delay is another issue that belongs in a Level 1 document.

When to Stop Transmitting

Another timer issue Franklin has brought up is a transmit turn-off delay (TX-Tail). He points out that some packet hardware can have many bytes of buffering, and although the host CPU may think a packet is done, the actual data may not have cleared the hardware yet. If the CPU then turns off the transmitter, the last packet will not have been completed, and the whole packet is therefore corrupted. He further points out that present implementations do not adjust this timing, but rather make it arbitrarily long to be sure the whole packet clears, regardless of channel speed. He suggests this timer also be user-settable.

Others indicate different methods can be used to insure the end of the packet has actually been sent. If a particular chip has a three-character buffer, the start of another frame plus three characters can be sent to the chip, then an abort command is issued to the chip, which will abort the short "timer frame", yet allow enough time for the last frame to be completely sent. This removes the necessity of having yet another special user-settable value which the user will not know how to set most likely. Implementers should take note of this trick in order to reduce channel overhead.

Additional AX.25 Level 2 Issues

In addition to the previous items, there are a few more that have come up whenever alterations to the AX.25 Level 2 protocol is discussed. These have not been thoroughly digested yet, and may be subjects for further enhancements at a later time. Some of them are listed below.

Parameter Negotiation

Several methods of negotiating different AX.25 operating parameters at connection setup have been discussed. The parameters most frequently talked about are packet data, and window size. Alteration of the default values have been suggested either by adding an information field to the SABM frame, or by using the XID frame (not presently allowed in X.25).

Either method violates the X.25 protocol standard, which does not seem to allow for such an option. Using I fields in connection requests (SABM frames) was done under the original Vancouver protocol. This seems to violate some basic frame rules, including Unnumbered and Supervisory frames not containing data (except for FRMR frames). The general feeling seems to be to not use this method.

Using XID frames to transfer special requests also has problems. Do the XID frames come before or after the connection is established? It has been suggested that an XID command frame with the Poll bit set be used to convey a special request, and an XID response frame with the Final bit set be used to indicate the response to the request. This seems more managable, and should be researched further.

Larger Window Sizes

When the Amateur community starts using satellites and high speed links for backbones, it may be beneficial to use larger frames and more of them. Our present limit of window size is seven, due to the three bit frame numbering plan implemented. Commercial satellite users have found that this limitation is too small when round-trip transmissions are in the order of a quarter-second for a geosynchronous satellite. The use of larger windows, by extending the numbering plan to seven bits, has allowed more effective use of satellites.

Extending the frame numbering to seven bits would allow up to 127 frames outstanding at a time. This would require a second control field byte. Use of the extended numbering should be considered in future AX.25 releases.

Selective Reject

While not allowed under X.25, other HDLC Level 2 protocols do allow the use of selective reject frames. These are used when a station receives a multiple frame transmission, with a bad one in the middle. The selective reject frame indicates not only that a bad frame was received, but gives the number of that frame, indicating it only needs that frame to complete the group. While the use of selective reject is not particularly time-saving with smaller windows, if the above extended windowing suggestion is implemented, some form of selective reject should also be implemented.

There is a possibility that many frames are missed, with selective rejects issued for each of those frames. This could actually perform slower than if all outstanding frames were retransmitted. Actual implementation should take this into account, relying on some mean value to determine which action to take.

Multi-Reject

Presently, once a Reject frame has been sent, another cannot be issued until the error condition has been cleared, Suggestions have been made that multiple rejects be allowed to be sent. simplifying error recovery. The exact operation of this with older devices is in question, and will also be subject to further study.

Data Compression

Franklin Antonio has also suggested that standard data compression techniques be implemented over AX.2.5 Level 2 links, to shorten transmission times. While this is a good idea, and should be further researched, it probably doesn't belong in the actual protocol specification.

Assignment of PIDS Based on Manufacturer

GLB Electronics requested a couple of years ago that certain Protocol IDentifiers be reserved and assigned to implementers. After much discussion over several meetings, this issue is still unresolved. The leaning of many at the moment is that this could be dangerous, potentially triggering user wars between system types. Many remain unconvinced that the benefits would outweigh the potential for harm.

Conclusion

This paper outlines most of the issues brought up regarding AX.25 Level 2 Version 2.0 since its adoption in October, 1984. There may be some additional ones that have been missed, which will be picked up at future and Conferences.

There have also been several suggestions and corrections to the text of the AX.25 Level 2 Version 2.0 document, which were left out for the sake of brevity. Most of these corrections have been indicated to the ARRL Digital Committee.

The next step is for this author to add the suggested changes to the AX.25 document and distribute the changed version to the Digital Committee, where it will be held under further review. At that time the SDL diagrams will also be added to the document.

After passing that review step, the Digital Committee will approve a final new version of the protocol, then have it printed and distributed.

Those with any comments, suggestions, or complaints should send them to this author at the above address. They will be passed to the Digital Committee in addition to being placed in the permanent AX.25 Documentation file kept by the author,which is the main basis for further AX.25 Level 2 modifications.

References

Lockhart, D.	"The Vancouver Amateur Digital Communications Group", Protocol Column, AMRAD Newsletter, AMRAD, June 1980
Fox, T.	"AX.25 Amateur Packet-Radio Link-Layer Protocol Version 2.0 October 1984", ARRL, 1984
Fox, T.	"Notes of East Coast Packet Radio Standards Meeting", Unpublished, January 23, 1982
Fox, T.	"Level 2 Protocol Proposal", AMRAD Newsletter, March 1982
Fox, T.	"Protocol Specification for Level 2 (link level) Version 1.0", Unpublished Draft, July 4, 1982
Fox, T.	"AX.25 Level 2 Protocol", Second ARRL Amateur Radio Computer Networking Conference, ARRL, 1983
Karn, P.	"Proposed Changes to AX.25 Level 2", Correspondence, June 1987
Karn, P. and Lloyd, B.	"Link Level Protocols Revisited", Fifth ARRL Amateur Radio Computer Networking Conference, ARRL, 1986
Antonito, F.	"AX.25 Proposed Changes", Correspondence, 1988
G3VPF	"Suggested AX.25 Specification Changes", The South-West AX.25 Group, 1988
Gustafson, E.	"Proposals for Updating AX.25 Level 2 Protocol Specification", Correspondence, 1988
Anderson, P.	"AX25L2V2", Correspondence, 1987
Johnson, L.	"Thoughts on AX.25 Level 2 Version 2.1", Correspondence, 1988
CCITT	"Recommendation X.25", CCITT, 1976, Ammended 1976, 1980, 1984
ISO	"Data Communications - Description of the 1984 X.25 LAPB - Compatible DTE Data Link Procedures", ISO Document 3179, ISO, 1984
Bell System	"Operations Systems Network Communications Protocol Specification BX.25 Issue 2", American Telephone and Telegraph Company, 1980
Bell System	"Operations Systems Network Communications Protocol Specification BX.25 Issue 3", American Telephone and Telegraph Company, 1983
ISO	"Data Communications - High Level Data Link Control Procedures - Frame Structure", ISO Document 3309-1976 (E), ISO, 1976
NBS	"Interface Between Data Terminal Equipment (DTE) and Data Circuit-Terminating Equipment (DCE) For Operation With Packet-Switched Data Communications Networks", Federal Standard 1041, Federal Information Processing Standards (FIPS) Publication 1041, National Bureau of Standards, 1983
BBN	"Defense Data Network X.25 Host Interface Specification', Defense Communications Agency, 1983
GSA	"Proposed American National Standard for Advanced Data Communication Control Procedures (ADCCP)", GSA, 1977

TRANSMISSION OF IP DATAGRAMS OVER NET/ROM NETWORKS

Daniel M. Frank, W9NK
1802 Keyes Avenue
Madison, WI 53711-2006

ABSTRACT

One of the main design goals of the Internet Protocol was that IP datagrams could be carried over existing local- and wide-area networks. This characteristic of IP makes it possible to build so called "internetworks" out of existing network facilities. We built support for an existing Amateur wide area network, NET/ROM, into the KA9Q TCP/IP package, allowing the use of NET/ROM to carry IP datagrams, and adding features which make the KA9Q software useful as a full duplex NET/ROM packet switch. We have also shown that NET/ROM may be used as a datagram network only, independent of its transport and application layer facilities.

INTRODUCTION

In the late Seventies and early Eighties, the world of computer communications consisted of many isolated local- and wide-area networks. Enough communications capacity existed to link the entire country, and much of the world, into a single large network, but the existing facilities were physically and logically dissimilar. They could not simply be "plugged together" to make this large "internetwork" (or "Internet").

The designers of the Internet Protocol (the "IP" in "TCP/IP") were committed to overcoming the obstacles that prevented an Internet from developing. They came up with two key ideas:

o **Gateways** can be established between networks. A gateway is a computer which possesses the physical resources and software to connect with and speak to more than one kind of network.

o A single protocol can be developed whose messages ("datagrams") can pass through any network, hidden inside that network's "native" messages. When a message with a datagram inside it encounters a gateway, the gateway "unwraps" the datagram, rewraps it in the native message of a second network, and sends it on its way. If a datagram is too large to fit inside the native message type of a network, the gateway breaks it into pieces ("fragments"), each of which is then wrapped in a native message and sent on.

By using IP and gateways, the designers of the Internet have created a global "network of networks", which today encompasses hundreds of thousands of computer systems, connected to every conceivable kind of network.

Amateur packet radio networking, like computer networking, consists of many different network technologies and protocols. Local AX.25 communications, digipeating, TexNet, ROSE, and NET/ROM, to name only a few, coexist or compete for dominance as the network technology of choice, This competition is healthy, and is in the spirit of amateur radio experimentation. Any attempt to establish one network over another as the single "standard" is both pointless and doomed to failure. No single standard can ever be imposed on radio amateurs any more than it could have been imposed on computer networks, given the investments already made in equipment, software, and education.

As the dissimilar amateur networks grow in size, they meet up with each other. Sometimes they coexist on the same channels. But without gateways and some kind of Internet Protocol, each network is an island of communication, unable to send or receive data beyond its own shores.

The work described in this paper is a first step towards true Amateur Radio internetworking. Using the KA9Q TCP/IP package as a basis, we have built a software system which functions as a gateway between local TCP/IP networks and the NET/ROM network. It allows IP datagrams to be forwarded automatically and transparently across existing NET/ROM facilities. In addition, as a full implementation of NET/ROM layer 3, it is capable of functioning as a NET/ROM relay node (as opposed to an AX.25 endpoint), and as **a** full duplex NET/ROM packet switch.

IP OPERATIONS OVER STANDARD AX.25 CONNECTIONS

In order to properly understand how we have interfaced to the NET/ROM network, we should first examine how "ordinary" TCP/IP operations take place over AX.25. This description follows the IS0 OSI Reference Model (RM), a seven-layer classification of network facilities.

From the bottom up, the layers used in packet TCP/IP operation are:

(1) A **physical layer,** made up of the radios, antennas, and modems used to generate and carry the tones used to convey digital data from one place to another.

(2) A **data link layer,** made up of HDLC and AX.25, used to format and address the data, detect errors and discard bad packets. The link layer only knows about and communicates with stations with which we are directly connected. In the case of packet radio, this means stations with which we have reliable, direct communications. (Digipeating doesn't count, for purposes of this discussion.)

(3) A network **layer,** responsible for routing packets to their destinations through one or more link-to-link hops. The main distinction between the data link and network layers is that the network layer provides facilities for communication between stations not directly connected. The network layer has to have some concept of **routing,** that is, the path to be taken by a packet to reach its destination. We use IP as our network protocol.

(4) A **transport layer,** responsible for reliable end-to-end communications. Our network layer does not guarantee that a packet will actually reach its destination. While AX.25 provides link layer acknowledgement and retransmission, it does not guard against nodes which go down, software errors, or a destination station which is not on the air. The transport layer provides for an acknowledgment to be sent from the packet's ultimate destination, and for retries in case that acknowledgment doesn't arrive within a reasonable amount of time.

The other function of a transport layer is **multiplexing.** The network layer provides only host-to-host addressing. However, a computer can have many users, and provide many different services. The transport layer takes incoming packets from the network and directs them to the proper programs based on information contained in the **transport header** portion of the packets.

Our transport protocol is TCP, the Transmission Control Protocol.

(5) **The session layer** is mainly involved with providing services to individual programs within the computer. It is not of importance for the current discussion.

(6) The presentation layer is mainly concerned with the uniform formatting of data, or its conversion between different character sets. Some of the TCP/IP user programs have a very simple presentation "layer" which maps plain text messages in the native character set of the user's computer, to and from ASCII with a standard line-ending convention.

(7) The application layer is made up of the various programs and services that use networking facilities. Users of TCP/IP mainly make use of **telnet,** for keyboard to keyboard chat and remote **login, smtp** for automated transfer of mail, and **ftp,** for easy exchange of files.

LINK LAYER MULTIPLEXING

As can be seen from our description, local TCP/IP operation uses regular AX.25 communications for its link layer. An AX.25 packet containing an IP datagram contains a special code in the protocol ID (PID) field of its header. This allows the link layer software to forward the contents of the packet to the proper part of the KA9Q package, in this case the IP routing code.

If the AX.25 packet contained a PID of "no level 3", the link layer would forward it to a different part of the package, in this case the AX.25 session code, which allows users of the package to hold "regular" AX.25 conversations, bypassing all layers between the link and application layers. (This brings up an important point about the reference model we've presented: an implementation may not contain certain layers from the RM if the services they would have provided are unused or unneeded.)

This switching of packets at the link layer based on their PIDs is known as **link layer multiplexing.** Multiplexing at the link layer is extremely useful, because it allows different network layer protocols to share the same data link services, and often the same link connections. Link layer multiplexing is what allows the KA9Q software with NET/ROM support to act as a digipeater, an IP relay, and a NET/ROM relay node, all on the same channel, through the same TNC.

AN OVERVIEW OF NET/ROM

Now that we understand how IP datagrams are carried over packet radio links, we should examine how NET/ROM operates. Again, we will use the IS0 OSI Reference Model as our framework:

(1) The **physical layer** is the same, i.e. radios, antennas, and modems.

(2) The data link layer is again AX.25, but the Protocol ID field of NET/ROM packets is set to a special NET/ROM ID.

(3) The network layer of NET/ROM handles the automatic routing of packets to their destination. A NET/ROM network packet header contains the source and destination callsigns of the NET/ROM endpoints. There is no information about the route the packet will travel to its destination. Instead, every node maintains a routing table based on routing **adjacencies:** it receives broadcasts from other nodes which say, essentially, "I am willing to take traffic for such-and-such a node." When a NET/ROM node receives a network packet, it examines its routing table to see if anyone is willing to pass it on toward its destination. If so, it hands off the packet to the next station. If not, it simply throws the packet away without comment.

The type of network communications service (as opposed to the routing techniques) used in NET/ROM (and IP) is usually called an **unreliable, connectionless datagram layer,** and the network layer packets are generally called **datagrams. The** service is **unreliable,** because it does not guarantee or confirm ultimate delivery. It is **connectionless,** because no **circuit** is established over which datagrams will travel. (This contrasts with some public data network protocols, where before data may be sent to a remote system, a fixed path to that system must be set up through the network, with resources preallocated at every intervening node. Each approach has its advantages and adherents.)

(4) The transport layer of NET/ROM uses what is called a sequenced packet protocol. Unlike TCP, which delivers an unsegmented stream of bytes to the receiver, and is free to pack as many or as few bytes into each message as it likes, the NET/ROM transport delivers a sequence of packets. The amount of data in these packets is determined by the amount of data in the AX.25 packets the NET/ROM user presents for transmission. NET/ROM is not free to combine packets together for greater efficiency, although it can fragment and reassemble packets' which are too large to fit in one of its transport messages.

The NET/ROM transport protocol provides end-to-end delivery and acknowledgement, as well as demultiplexing of arriving messages by circuit number. A NET/ROM node can be handling traffic for more than one circuit, or connection, at a time, and it directs that traffic internally by examining the circuit number field of the transport header.

(5) The session layer is not present in NET/ROM.

(6) The presentation layer is not present in NET/ROM.

(7) The application layer is what a user sees when he or she connects to a NET/ROM node. It is responsible for responding to user commands to list routes and nodes, and establish connections. "No layer 3" AX.25 packets arriving at a NET/ROM node are shunted directly up to the application layer, while "NET/ROM" PID packets are forwarded up to the NET/ROM network layer. This link layer multiplexing should be familiar from our earlier discussion.

A full explanation of how the NET/ROM software works is beyond the scope of this paper. The reader is referred to the NET/ROM manual for further details.

NETWORK AND INTERNETWORK

Our presentation of the IS0 OSI RM has been somewhat simplified. In particular, the IS0 recognizes a subdivision of Layer 3 into a Network Layer (3A) and an Internetwork Layer (3B). Strictly speaking, the Internet Protocol (IP) is a 3B protocol, while the NET/ROM network service is a 3A protocol. To put it somewhat crudely, IP is an Internetwork Layer because its messages can be routed through multiple logically and physically distinct networks. The same cannot be said of X.25, for example, or of NET/ROM layer 3. Our NET/ROM support in the KA9Q package reflects this distinction.

The KA9Q NET/ROM software is not a full NET/ROM implementation. That was unnecessary for our purposes. We didn't need the NET/ROM transport protocol, since our reliable end-to-end services are already provided by TCP. We didn't need the application layer, for similar reasons. What we did need was an existing network service that could carry our IP datagram traffic to remote destinations simply and easily. The NET/ROM network layer was sufficient for this purpose.

We use the NET/ROM nodes as a datagram network. When we have traffic to pass through a local NET/ROM, our software makes sure we have an AX.25 connection to that node, then puts a NET/ROM layer 3 header on our IP datagram and sends it off to the NET/ROM via an AX.25 packet with a protocol ID of NET/ROM. The NET/ROM link layer sees the protocol ID and passes the packet to its network layer, which examines its routing table and passes the packet on to the appropriate neighboring NET/ROM. This process continues until the packet arrives at the destination computer running the KA9Q software, where it is unwrapped and passed back up to the IP code.

At no point is the NET/ROM user interface or transport layer involved. We do not have to issue CONNECT commands, or make use of NET/ROM virtual circuits in any way. The NET/ROM nodes accept and pass our datagrams because they do not examine the contents of network datagrams not specifically addressed to them. We are able to take advantage of the link-layer acknowledgements and automatic routing of the NET/ROM system without the overhead of its higher level services.

SOFTWARE ARCHITECTURE

Let's examine how this is done in more detail. Our once-simple protocol stack has grown a bit by now. Let's have a look at it:

(1) The physical layer is basically unchanged (although we did added another physical layer service, described later).

(2) At the data link layer, we still have AX.25. However, the link layer now multiplexes three different kinds of packets, "No level 3" packets still go up the AX.25 session code, and IP packets will go directly up to layer 3B, but now we also direct packets with a NET/ROM PID to the NET/ROM 3A routing layer.

(3A) Incoming packets with a NET/ROM PID go to the network layer. This is a full implementation of NET/ROM layer 3. It has its own routing table, similar to that found in any NET/ROM node. It sends NODES broadcasts, which update the routing tables of neighboring NET/ROM nodes, and updates its own routing table on receipt of NODES broadcasts from those neighbors.

The NET/ROM layer examines incoming NET/ROM datagrams to see if our station is their destination. If the datagrams are not for us, the routing table is examined to see if we can forward them on to a neighboring node for handling. If we can, they are sent back down to the link layer to continue their journey. In other words, a station running the KA9Q package with NET/ROM support can act as a NET/ROM relay station. As far as neighboring NET/ROM nodes are concerned, they are simply passing traffic on through another NET/ROM.

If a NET/ROM datagram is for us, the network layer makes sure that it isn't a NET/ROM transport packet. If it is, it is dropped. If it isn't, it is sent up to layer 3B.

(3B) The intemetwork layer contains the IP router and protocol code. (Remember that IP has its own routing table and algorithms!). It receives AX.25 traffic with a protocol ID of IP, as well as IP datagrams arriving in NET/ROM network datagrams.

The remaining layers are the same as before, so we won't repeat them.

IP ROUTING VIA NET/ROM

The IP routing table is similar, although not identical, to the one used for NET/ROM. It contains two kinds of entries, which we will call local routes and gateway routes.

A local route consists of an IP address and an interface name. The KA9Q software supports multiple interfaces, similar to the way that NET/ROM supports both a TNC's modem and its serial port. One component of a route, both in NET/ROM and the KA9Q package, is the interface through which an outgoing datagram should pass. (The main difference is that, while NET/ROM only supports AX.25 and two interfaces, the KA9Q code supports many different link layers and an almost unlimited number of interfaces.) When a local route is found in the IP routing table, this means that the station with the given IP address is on the local subnet, which for packet radio purposes means that it is within radio communications range. The datagram is forwarded to the link layer with an indication that direct delivery should be attempted.

A gatevay route consists of an IP address, an interface name, and a gateway IP address. When we encounter a gateway route, it means that the station in question is not on our local subnet (i.e. not within radio range), and must be reached via a relay station, or gateway. (You will recall the idea of gateways from our earlier introduction of internetworking.) The IP datagram is forwarded to the link layer with an indication that the message should be sent, not directly to its destination, but to the gateway station, which will make an attempt to reach the recipient, perhaps via another gateway.

When we added the NET/ROM support, we were concerned that it be fully transparent to the IP layer, both out of concern over proper design, and out of a desire to avoid any unnecessary rewrite of the. existing code. One key assumption in the KA9Q IP routing software is that of routing adjacency: the IP layer makes the assumption that it can reach, via the interface given, some IP address mentioned in the route entry (either the recipient or the gateway), However, we are using NET/ROM precisely because there is no IP station within radio range who can handle our traffic. In order to maintain the adjacency assumption at the IP layer, we had to simulate the presence of an adjacent IP station in the NET/ROM code.

An IP route which uses the NET/ROM support looks just like any other routing table entry: it consists of a destination, an interface, and an optional gateway. The only difference is that the interface is called "Netrom", and it's not a link layer interface at all, although it appears that way to the IP routing code. When the IP layer sends a datagram down to the NET/ROM "interface" for handling, it is actually calling a small stub routine above the NET/ROM routing code. This stub looks up the IP address in a table which associates IP addresses, used at the IP layer and above, with AX.25 callsigns, used by NET/ROM% network layer. If it finds an entry for the given IP address, it creates a NET/ROM network layer datagram header with a destination address set to the AX.25 callsign found in the association table, prepends this header to the IP datagram, and hands it off to the NET/ROM routing code.

The NET/ROM routing software now handles the datagram exactly as it would any NET/ROM traffic coming in from outside: it checks to see if there is an entry for the destination AX.25 callsign in its routing table. If there is, it opens a link layer (AX.25) connection to the neighboring NET/ROM node advertising the best quality route, and forwards the message into the NET/ROM network, to be delivered (ultimately) to the station whose AX.25 address is that in the destination field of the NET/ROM network header, and whose IP address was that of the destination or gateway in the IP routing table entry.

This approach was extremely successful. Not one line of code needed to be changed in the IP routing code of the original KA9Q package.

FEATURES TO SUPPORT NET/ROM PACKET SWITCHING

As implied above, the KA9Q package allows an almost unlimited number of interfaces to be used for receiving and forwarding packets. At the IP layer, datagrams are routed from interface to interface using information from the IP routing table. We added a similar functionality to the NET/ROM routing layer, allowing it to receive and send NET/ROM traffic on any AX.25 interface it is configured to use. This feature allows the KA9Q software with NET/ROM support to be used as a multi-port full duplex NET/ROM packet switch, using standard TNCs or modem boards available for the IBM PC. This is vastly superior to the practice of wiring together several NET/ROM TNCs with a diode bridge. There is no possibility of collisions, since each TNC has its own serial port or bus address, so the interfaces can all run at full duplex, full speed through the switch. In addition, this arrangement can be used with the high-speed interfaces and modems now becoming available, far exceeding the capabilities of a standard TNC.

In several places in the United States, excess bandwidth on commercial data links is being used to carry NET/ROM traffic. These "wormhole" links work fairly well when there is only one NET/ROM at each end, but their performance degrades quickly if more are added. Beyond the difficulties inherent in the diode bridging scheme shown in the NET/ROM manual, there is an additional problem not amenable to a simple hardware solution. Most of the data links being used are running through time-domain or statistical multiplexing hardware, or through public data networks. While all of these provide some kind of carrier detect indication, in almost every case that indication comes far too late to avoid collisions. Carrier sense simply doesn't work, since the carrier indication isn't there when it is needed, and arrives just in time to cause unnecessary delays afterwards. Performance of such an arrangement is likely to degrade to below that of schemes using no carrier sense at all.

NET/ROM nodes use a simple serial framing method to communicate with each other over their serial ports. We have added support for this framing method alongside the "KISS" protocol which the KA9Q package normally uses to communicate with TNCs. It is possible to plug a number of NET/ROM TNCs directly into the serial ports of an IBM PC, and use the PC as a switch. Some of those NET/ROM TNCs can be at the ends of "wormhole" links. These links can run at full duplex with no collisions, thus getting maximum performance and almost zero retries (assuming reliable data lines and serial interface hardware). The NET/ROM serial interface code is instrumented to provide statistics on traffic volume and error rates on its serial ports.

LESSONS LEARNED

The creation of the NET/ROM code has provided some interesting lessons on how we should and should not go about building amateur packet networks. One of these lessons became apparent before we even thought of writing the NET/ROM code.

We would probably not even have added the NET/ROM layer three support to the KA9Q package had there been an easier way to accomplish what we wanted, which was to use NET/ROM networks to handle our IP traffic until we could build our own IP network. Unfortunately, the NET/ROM software has a rather unfriendly link layer multiplexor. It sends "no layer 3" packets to the application layer, and "NET/ROM" packets to the NET/ROM network layer, but anything else it consigns to oblivion. We could have built fairly simple code to establish connections and send our IP traffic over NET/ROM transport circuits, but any packet with a protocol ID of "IP" was simply dropped by the NET/ROM software. So, lesson number one:

If you're going to build a networking product, write the multiplexing code to be inclusive, rather than exclusive. In other words, if you get something with an unfamiliar protocol ID, wrap it up and send it on, remembering to regenerate the PID properly on the other end.

Another problem we encountered was the lack of a protocol ID field in the NET/ROM network layer header. Both AX.25 packet headers and IP datagram headers contain a field which indicates what sort of higher level protocol stuff is packaged inside. This is not unlike the cans on your grocer's shelf: without a label, you have a hard time telling the beets from the beans. The AX.25 protocol ID field makes link layer multiplexing possible, and the protocol ID of an IP datagram header allows many higher level protocols to use its internetworking services. Because the NET/ROM network header does not contain a protocol ID field, there is no straightforward way to put anything but a NET/ROM transport packet inside. This is unwise. The authors of NET/ROM may well have been unaware that anyone would ever attempt a project such as ours, but by leaving this feature out of their network layer, they made it difficult, if not impossible, to ever introduce other transport protocols into their product line. So, lesson number two:

Include a protocol ID field in your link and network layer headers, even if you can't think of a use for it yet. Make it big enough to be useful, and offer to be the repository of assigned PIDs, so that a standard develops.

In experimenting with the auto-routing code in our NET/ROM network implementation, we discovered something that is a common complaint among NET/ROM operators. This can be summed up by the dictum, "Just because you can hear them, doesn't mean they can hear you." It is not unusual to have neighboring nodes that are "alligators" (big mouth, tiny ears) or for your node to be a "rabbit" (big ears, tiny mouth). Also, band openings on two meters happen quite often, and usually last just long enough for you to receive a routing broadcast from a station from whom you will never hear again - at least until the next band opening. Either situation leaves your routing table cluttered with impossible routes, which can lead to repeated link-layer retries and transport layer failures. Routes based on band openings age out fairly quickly. Ones based on deaf neighbors come back, again and again.

After a bit of experience with this phenomenon, we added the "nodefilter" feature to our implementation. The user may specify a list of nodes which are the only ones from which route broadcasts will be accepted, or alternately, may specify a "reject list" of nodes whose broadcasts will be routinely ignored. The lesson:

If your routing method involves broadcast in an asymmetrical or inconsistent communications environment, provide a way to restrict the routes accepted to those offered by reliable nodes.

The implementation described was actually the second one we did. The first one grabbed AX.25 interfaces away from the IP part of the KA9Q code, and could only be used for NET/ROM and regular AX.25 traffic. This appeared to be a horrible idea from the moment the first version was completed, and prompted an immediate rewrite, producing a program that could act as a packet switch for IP as well as NET/ROM. The lesson here (besides "look before you leap") is:

If you're going to build a packet switch for amateur use, support link layer multiplexing, and try to make it multi-purpose, This is a hobby, and the radios, tower space, and dollars are in short supply. The more stuff you can do with a single one of each, the happier you will be in the long run.

EXPERIENCE

At this writing, experience with the software is necessarily limited. It is only now being made an official part of the official KA9Q release (thanks, Phil!), so it has not been widely available as for as long as we would have wished.

Still, it has found its way into enough hands for us to have some preliminary measurements and impressions.

The Madison, Wisconsin NET/ROM node (MAD) is connected to a node (MQTA) in Marquette, in the Upper Peninsula of Michigan, via a multiplexed commercial data line. We have conducted tests between W9NK, in Madison, and KV9P, in Alpha, Michigan. Both stations sent periodic routing broadcasts to announce their presence to their local nodes, The path chosen by NET/ROM was:

W9NK <-> MAD <-> MQT <-> MQTA <-> IRN <-> KV9P

where MQTA and IRN were NET/ROM or TheNet nodes in Marquette and Iron Mountain, Michigan, respectively, All nodes except the two using the data line were on two meters, with a speed of 1200 baud.

Performance was surprisingly good, with TCP round trip times settling in around 12 to 20 seconds, with a standard deviation of about nine seconds. In spite of the number of hops, performance was good enough to hold fairly coherent keyboard-to-keyboard conversations.

We did note at least one case where duplicate copies of datagrams were delivered by the NET/ROM network. Since TCP discards duplicates, this causes no problem in normal operations, but in the case we noticed it resulted in two replies to the same ICMP Echo Request message (produced by the ping command).

Feedback from other users, particularly NOAN in Iowa, illuminates a serious problem with the management of existing NET/ROM networks: the routing tables of these networks are so inaccurate that many experienced users don't use the network layer facilities at all! BBS mail forwarding scripts are set up to establish connections to the local NET/ROM node, request a transport connection to a selected neighbor, then from that neighbor to another, and so forth to the NET/ROM node in their destination area. They have discovered that, without human intervention, many NET/ROM networks* routing facilities break down and become unusable.

This problem has some impact on normal operations, in the sense that these multiple transport sessions do not in any sense add up to end-to-end protocol support. There is in fact no transport facility (as we understand the term) in use in these cases, since no acknowledgements travel from one end of the communications path to another. There may as well be no transport layer in NET/ROM under these circumstances; the overhead would at least be substantially lower, with no additional loss of reliability.

Unfortunately, networks in such a pathological state are unusable by our TCP/IP software. Since we make no use of the NET/ROM transport layer, we must rely entirely on the accuracy of the network layer routing tables to support the forwarding of our packets to their destinations. If these tables are not correct, our traffic will not get through.

The good news is that, in some areas where the NET/ROM operators are also working with TCP/IP, this problem is forcing them to pay attention to the quality of their routing tables. As we have noted above, NET/ROM is somewhat short on facilities to do this, but a few things can be and are being done with the tools available. One side-effect of the TCP/IP NET/ROM support may be an improvement in quality of service to all NET/ROM users!

FUTURE DEVELOPMENT

We hope, at some point, to produce a version of this support that can be put into ROM and used in a dedicated packet switch for hostile environments. Such a switch would allow us to begin building IP networks, while also offering superior performance to the NET/ROM community. It is our hope that the two user communities can work together, sharing resources to build a better network than either could alone.

ACKNOWLEDGEMENTS

Sincere thanks are due to Phil Karn, KA9Q, who would have been rich (or at least, richer) by now if he hadn't been dedicated to improving the state of the art for all radio amateurs. Also, thanks to everyone on the tcp-group mailing list, who contributed helpful comments during the design and development of the NET/ROM code. Special thanks also to Howard Leadmon, WB3FFV, and John Limpert, N3DMC, who got it to compile under Unix.

Individuals who helped test or provided feedback on experience with the software were: Duane Brummel, NX9K; Hasan Schiers, NOAN; and Dave Reinhart, KV9P.

Phil Karn, KA9Q, reviewed the first draft of this paper, providing many helpful comments and suggestions.

MORE AND FASTER BITS
A Look At Packet Radio's Future

Bdale Garbee, N3EUA

Hewlett-Packard Colorado Springs Division

ABSTRACT

The biggest problem facing amateur packet radio today is the inability of the ham community to envision the breadth of possibilities that exist once higher speed modems become available. This paper attempts to survey some of the applications popular in other networking environments, and comments on their possible use in the amateur service. In addition, a preliminary report on work in progress to develop multi-megabit per second connections on the microwave bands is presented.

1. Background

Much of the effort expended to develop the current amateur packet radio network has come from people whose primary orientation is towards radio and *'communications", and not towards formal networking, or even "computer communications". While we owe a debt of gratitude for the efforts and commendable successes of this group, it should be equally obvious that we need to take a 'networking view" of our future if we are to successfully advance the state of the art in amateur packet radio.

Even in the world of professional networking, there are many different application scenarios, each involving a different set of priorities and solutions. We must be very careful to make sure that decisions we make about protocols are reasonable given the environment in which we operate. Because our environment is unique, some techniques, benefits, and restrictions of wired networks do not apply to us. In addition, we need to be aware that sometimes a solution used in one kind of wired network applies closely to our needs, while at other times another kind of wired network more closely resembles our situation. More about this later.

It is unfortunate that the evolution of grass-roots BBS networks such as Fidonet have involved as many false starts and wrong turns as they have. This is particularly true when we consider that these efforts have in many ways paralleled the early efforts of such seminal groups as the Arpanet research community. Here, as in amateur radio, it is unfortunate that the movers and shakers are primarily *not* networking people, and therefore do not have at their disposal the set of historic knowledge and lore that would have allowed them to avoid many of the pitfalls they have encountered.

What impact does this have on us in the amateur packet world? Merely that we should not be afraid to evaluate and adopt existing hardware and protocol standards *when* and *where* they make sense. Lets *not* fall into the "Not Invented Here" trap, dooming ourselves to repeat all the mistakes of those who have come before us.

We can, and should, do it better in the amateur world. What we need is to expand our vision of the future to include applications and technologies that may not at first seem particularly relevant. In order to do this, we need to know what has already been done, and build from there. This paper attempts to provide an overview of applications, and some incentive for further study. The rest is up to you !

2. Application Overview and Bandwidth Requirements

In the world of wired networks, it is common to segregate network applications into two categories: those that require noticeable bandwidth, and those that do not. The term "noticeable" is wonderfully vague, and has come into wide use as a result of the fact that the base bandwidth available is fairly high, typically 10Mbits/sec. Using Ethernet, it is not uncommon to find an entire software development lab running Telnet and SMTP-based mail on a single shared cable with little degradation in response time.

In the amateur packet radio world, we are currently somewhat less fortunate. The defacto standard data rate is 1200bits/sec half duplex. This is over 800 times slower than coaxial lans at 10Mbit/sec. It is therefore not surprising that we should be substantially more concerned about available networking bandwidth on our RF channels than most users are on wired networks. In order to better understand why we need faster data rates, let's take a look at some existing applications, and make some qualitative assessment of their characteristics.

Electronic *mail* is wonderful from a networking standpoint, because it is typically both a low bandwidth application, and a fully non-real-time application. By this, we mean merely that the volume of electronic mail generated by an average user it typically very small with respect to the available network bandwidth, and that the user in a rational electronic mail scenario is able to locally create and queue mail, and thus is not constrained to wait in real time for a mail transfer to occur. The computer takes care of it in the background. From this we can conclude that electronic mail is unlikely to be a burden on network bandwidth. The high degree of success achieved by the current PBBS mail forwarding network, using only 300 baud HF and 1200 baud VHF link, supports this conclusion.

The use of a *virtual terminal* protocol is at almost the exact opposite extreme. Here, the essence of the application is real-time use of a remote computer by a user as if on a local terminal. In this case, the response time to individual keystrokes typed on the keyboard is the limiting factor. We have all experienced the frustration of trying to access a remote PBBS system, and the agonizing delays associated with waiting for the system to "come back". This is therefore a class of application which we should consider supplanting with different ways of operating (local mail editting and queueing as opposed to online entry of mail text into a PBBS, for example). If we really need to continue supporting virtual terminal services, and I believe we do, then this may be an ideal application of much higher speed modems. Not because the volume of data we wish to transfer is high, but because of the need for fast round trip times, and minimized channel access delays.

File transfer protocols fall sornewhere in between mail and virtual terminal access. This is due in part to the possibility of using either an FTP-like real-time user interface, or a background batched mode of operation. An interactive interface to a file transfer makes the user aware of the time involved to complete the transfer, and therefore the network bandwidth. In this case, it is the need for the network to move large blocks of data quickly that causes a desire for more bandwidth, not necessarily a need for short round trip times. On the other hand, a batched, background file transfer protocol allows the user to specify transfer parameters, and then proceeds to completion without the need for user level intervention. If our use of file transfer protocols is primarily real-time, we care a great deal about the speed at which they occur. If they are primarily background tasks, we are likely to be less concerned, since we can go do other things while the transfer takes place. Thus, file transfer is an application where our need for speed can be very high, depending on our usage patterns.

There exists an entire class of protocols which have not to date been used by the amateur radio community, doubtless because their data transfer requirements are so high as to make their use on a 1200 baud network essentially impossible. Perhaps the most well known of these protocols in the wired network world is NFS, the Network File System. In this application, a file system on one computer can be made available to another computer system transparently. A user on machine A can be given full operating system level access to a storage device on machine B as if it were directly attached to his own system. This is a natural evolution of, and very desireable replacement for, simple file transfer protocols in a homogeneous computing environment.

When we talk about local file repositories in the amateur world, we tend to mean PBBS's with the ability to 'upload" and "download" files, or the idea of an FTP server with massive amounts of disk space available for storing files. I would like to propose that we instead try to consider the notion of a local NFS server, making available sets of related files as file systems, allowing a multitude of varying access mechanisms based on need. In order to make this work, we need *much* higher speed networks. This is because NFS in essence replaces a local disk and controller, and so is expected by the user to perform at rates approaching those of fixed disk controllers.

If it is difficult to imagine using NFS in an amatuer environment, substitute any of a number of other exciting high bandwidth-requirement applications (digital voice, digital video, etc), and the analysis will be very similar.

3. So, Where Do We Need Speed?

Given the preceding brief evaluation of existing high-usage protocols, what should our direction be with respect to application of higher data rates? The traditional model of "the network" assumed in the amateur packet radio community seems to be one of high speed backbones connecting lower speed local area networks. This is in direct opposition to the existing model in use in the wired network world, leading one to wonder if it is correct.

In a typical corporate or university network, fairly high speed local area networks (LANs) are built within a functional area or building, typically using Ethernet technology at 10 Mbits/sec. When it becomes desireable or necessary to link these LANs toegether, gateways are installed which operate at the full LAN speed on one port, and at some reduced speed on the linking port. Typical data rates connecting these gateways are those available from commercial com m on carriers. *2400* or 9600 baud via dialed phone lines, 56kbits/sec or approximately 1.5Mbits/sec via leased lines, and so. obviously, these data rates are much lower than the LANs, leading us to recognize a model of high speed LANs connected to low speed backbones. Why the discrepancy with amateur packet radio's model? To answer this, we need to look at how the previously mentioned protocols are typically used in a working environment.

Mail is equally likely to be sent to an associate on the local LAN cable, or an associate at a remote site, perhaps with a tendancy towards higher local usage in environments where electronic mail is used as a primary communications tool within a working group. Regardless, since it is a relatively low network bandwidth application, the impact on required speed is very low.

Virtual terminal access is interesting because while it tends to not produce very large quantities of data transferred, there is a strong tie to perceived response time at the user level. Thus, we may require very high available network bandwidths in order to produce short round trip times, but not be using a very large percentage of that available bandwidth. This is the classic example of an application where speed is an issue not because of the volume of data to be transferred, but because of the need to transfer small pieces of data very quickly. Virtual terminal sessions are typically weighted very heavily towards local use, since a programmer might have access to several local machines within his group, but only infrequently need to log in to a computer system at a remote site.

File transfers are heavily weighted in actual practice towards local usage. A typical software engineer working in a corporate R&D environment might use several different computer systems to accomplish his work, frequently moving files back and forth between the various local systems. That same engineer might find it useful on occasion to obtain a piece of software or a document from a remote division of the company, or from some other remote site. While it is difficult to characterize the actual split between local and remote file transfers, it is fairly obvious that more bandwidth will be needed locally than on the connecting backbone.

The NFS protocol draws both virtual terminal and file transfer concerns. Because it serves as a virtual disk drive interface, users are apt to expect it to perform with a response time similar to that of convential fixed disk drive interfaces. In addition, since it is frequently used to load executable programs and/or data files from a remote disk subsystem, the amount of data transferred can be very large over short periods of time. Thus, in order to make use of protocols like NFS, packet voice, or packet video, we need ***lots*** of bits per second locally. The remote use of NFS is almost non-existent.

It appears that those discussing high speed backbones for amateur radio are for whatever reason anticipating that almost all traffic will pass out of the local LAN. This is without precedent in the wired network world, leading one to suspect that it may not be an accurate assumption.

4. Faster Modems on the Market

As mentioned earlier, the defacto data rate in amateur packet radio is 1200 bits/sec. Some time back Kantronics introduced a 2400 bit/sec modem option on their TNC's, but this has been slow to catch on probably because a factor of two increase in data rate,is more than offset by the non-standard nature of the modem, concern over connection with existing stations.

Work done by K9NG and others has resulted in the availability of 9600 and 19200 bits/sec modems, but these have not come into wide use as a result of the need to modify radios in order to work effectively at these speeds. This situation may change if TAPR follows through with its plans to develop RF transmit and receive modules specifically for use with K9NG-style modems.

Commercial vendors such as GLB and AEA also produce 9600/19200 baud modem-s, complete with radio gear. Unfortunately, the relatively high cost of these unit, coupled with the recently justified uncertainties about the future of the 220Mhz band, have prevented their widespread use.

A year ago, WA4DSY published his design for a 56kbit modem, which has met with great enthusiasm. However, the cost of a fully configured WA4DSY modem including RF gear and antenna can easily top $700, and there is very little available in the way of digital hardware to drive the modems at 56kbits.

5. A Project in Progress

Recognizing the need to develop much higher speed data transmission technology in order to expand the base of available applications for packet radio, Glenn Elmore, N6GN, and I have begun a project to integrate Ethernet controller chip technology with microwave transceivers. The eventual goal is to produce a unit which attaches directly to an off-the-shelf Ethernet controller card via the 15-pin connector, and which terminates in perhaps a 4-foot dish at 10Ghz or 24Ghz on the RF end.

The immediate advantage of this approach is the ubiquity of the 15-pin Ethernet connector, which appears on everything from PC plug-in controller cards to high-end engineering workstations and mainframes.

Projected cost of each unit is about half of a fully configured WA4DSY 56kbit station, providing 10Mbit/sec data rates point to point. Because many Ethernet controllers are already supported by software such as the KA9Q TCP/IP package, these units will be immediately useful without the need for software development, which has been one of the problems with digital interfaces for the WA4DSY design.

While the characteristics of microwave transmission force some restrictions on the application of this concept, we feel that it will be a substantial boost to application developers to have this kind of data rate available.

6. Issues at Microwave Frequencies

6.1. Directivity

In the wired network world, there tend to be many networking hardware technologies in use, but they can typically be broken into two distinct categories. Some are busses such as ethernet which may contain large numbers of hosts sharing a single connection media, with the resulting ability to broadcast, and the need to share available bandwidth. Others are point-to-point links, which typically connect two systems only, with no broadcast capability, but with the full channel capacity available for each link.

Modems such as the K9NG and WA4DSY designs are typically operated on the VHF and UHF bands, where omnidirectional antennas are quite feasible. Because of this, they are ideally suited to broadcast-oriented network design, and will continue to be the best solutions where many stations need to be able to communicate with a central repeater site.

At the microwave frequencies required for truly high-speed data transfer, improving signal fidelity usually involves increasing the gain, and therefore directionality, of the antenna system. This is because of the relative difficulty of generating large transmitter output powers at microwave frequencies. The extreme directivity of this approach, coupled with the line-of-site nature of microwave communications, means that point-to-point network topologies are necessary. This is ideal for backbone links, since they are typically "mountaintop to mountaintop" anyway, and we get the added benefit that local users can't interfere with the backbone since the line of sight path is typically beyond their reach. It is also satisfactory for specific high-bandwidth requirement local links, such as between two users involved in packet video experimentation.

6.2. Addressing

In order to make this project economically feasible, it is necessary for us to base the digital interface design on the readily available, fairly inexpensive VLSI Ethernet interface chip sets available from National Semiconductor and others. The result of this is that we will be forced to use the 48-bit Ethernet address and packet framing format as the basic bit protocol in this project. Widespread application of this technology may dictate a change in the FCC rules to allow non-AX.25 addressing in packets transmitted on amateur radio. Gross hacks are no doubt possible that would allow some encoding of the user callsign to serve as the 48-bit address, but this seems both counter-intuitive and counter-productive, in part because it might entail modification of host networking software.

There is an existing standard for the issuance of Ethernet addresses that causes them to be tied to specific hardware. This should allow them to serve as unique identification for monitoring purposes, assuming that FCC or volunteer monitoring of a 10Mbit signal on 24Ghz is even possible.

6.3. Data Rates

While our initial experiments are targetted at data transfer rates of 500kbits/sec to 2Mbits/sec, development of a digital transceiver capable of 10Mbits/sec that will function over a 100 mile line-of-sight path seems feasible. An initial prototype at the lower data rate may in fact be operational by the time this paper is published. In any case, we hope to successfully complete the project sometime in the next year.

7. Conclusion

The ham community has embraced a set of applications on packet radio that mirror the capabilities of other familiar modes, and other similar services such as the telephone BBS network. If we want to remain on the cutting edge of technology, thereby eliminating the now-too-common "packet burnout" syndrome, we need to expand our horizons to encompass applications that will soon be possible with advances happening *right* now in the packet community.

The earlier qualitative look at bandwidth utilization on wired networks, backed up by some quantitative analysis of the HP Corporate Internet, leads to the conclusion that we need lots of bandwidth locally, and progressively less bandwidth as we get farther and farther away in a networking gateway sense. On the other hand, hams are notorious for wanting to participate in personal DX, and therefore we may want more than a mere minimum data rate on our long haul networks. However, we should concentrate on our local networks, because that's where the bulk of our utilization will be, and because it's going to be easier to rationalize spending big bucks on hardware that we use personally every day than it will be to rationalize big bucks on mountaintops.

For the majority of our local network activity, modems of the K9NG and/or WA4DSY variety are probably* the best choice. This is primarily because they operate on bands where omnidirectional antennas are possible, supporting the idea of multiple-access repeater and central facility sites without the need for lots of duplicate hardware. In order to make these units practical, we need to develop some easily duplicated, low cost RF modules. We do not, as yet, have a "complete solution."

We may soon have available RF gear that plugs into an Ethernet controller and provides lOMbit/sec data transfer on line-of-sight, point-to-point links. This may be the ideal technology for backbone links, and will probably also find application in local nets housing "power users", much as the 386 AT-clones have infiltrated corporate environ men ts largely dom inated by slower speed machines.

So, what will our future network look like? Why don't we try for 9600 baud "average ham" hardware based on K9NG modems and the forthcoming TAPR RF hardware, 56kbit local capability for power users, 10Mbit microwave for the esoteric high-bandwidth local applications and backbone links, and an applications mix that includes remote file access, digital voice, and digital fast scan TV! Anything is possible, with enough bits!

8. Acknowledgements

The material in this paper has been influenced by several individuals. First and foremost is Glenn Elmore, N6GN, of the Hewlett Packard Network Measurement Division. Glenn is developing the actual RF/Microwave hardware that we hope to run at 10Mbits/sec in the near future. Mike Chepponis, K3MC, has demonstrated considerable insight about the possible futures of amateur packet radio during our electronic mail discussions of the last year or so. Thanks also to Bob Gobrick, WA6ERB, and the many members of RMPRA who provided input at the 1988 RMPRA PacketFest about the kinds of applications that they hope to see running on packet radio in the future.

9. References

Goode, S., *Modifying the Hamtronics EM-5 for 9600 BPS Packet Operation*, Fourth Computer Networking Conference, 1985.

Heatherington, D., A *56 Kilobaud RF Modem*, Sixth Computer Networking Conference, 1987.

Ingram, D., *An Expandable Microwave Network for Multimode Commun ica tions*, First Computer Networking Conference, 198 1.

Tannenbaum, A., *Computer Networks*, Prentice-Hall, 1981.

National Semiconductor, *Series 32000 Databook*, Data Sheets on the DP839X Ethernet Controller Set, 1986.

Can We Continue To Ignore Level One?

by
Eric S. Gustafson, N7CL

ABSTRACT

For some reason which I cannot fathom. there has been a great reluctance to specify or even to provide guidelines for the various level 1 issues in the amateur packet radio system. This reluctance traces back a22 the way to the very early days of packet radio development. I find this situation very strange indeed since if level 1 isn't working, all the other levels of the protocol which everyone seems eager to specify down to the last bit position are all irrelevant.

In this paper I will choose one of the most consistently botched and yet most easily corrected leve2 1 parameters, the modem's data carrier detector. I will show how the performance of our present packet systems can be improved by careful consideration of even just this single level 1 issue.

Data Carrier Detect (DCD) is one of the most important items to consider on any mu2 tiple access packet channel. This paper explains why, then describes how to improve the performance of the current generation of amateur packet terminal node controllers (TNC).

INTRODUCTION

It ha8 been six years now since the birth of the AX25 level 2 standard in late 1982. I think that now it is appropriate to ask "What has the ostrich approach to setting level 1 standards gotten us so far?" Well, it has gotten us a number of things:

1. It has gotten us a user base that is almost completely ignorant that there is anything that needs to be considered at level 1. This is too bad since the user is the only one who has any control over what happens there.

2. It has gotten us manufacturers building TNCs with modem8 that have a number of characteristics which are detrimenta to the performance of what wa8 originally touted as a SHARED channel.

3. It has gotten us radio manufacturers who are under zero pressure to provide product8 which physically interface in a standard manner or even use similar levels for the digital mode audio signals. Manufacturers who are still content for the most part to build radios which are from 3 to 10 (or even more) times as slow a8 necessary for good packet system performance (NO aspect of voice radio performance must be compromised in order to achieve adequate speed).

4. It has gotten us a protocol specification which so judiciously avoids any provision for any level 1 issue whatsoever that its performance is significantly degraded when used on a channel where the signal propagation is anything less than similar to a 1 and l ine. This could probably be adequately fixed as an implementation issue but there is NO information available to the implementer warning him of the necessity.

Do I blame the manufacturer8 of radio8 or TNCs? No not at all. The manufacturers have no reason to gamble on trying to set up anything in a standard manner un2ess the standard pre-exists and there is some consumer pressure on a21 to bring about general conformance. With relatively short product lifetimes and low profit margins in the amateur market nobody could expect them to spend a lot of engineering time to optimize the modems, for example, before getting a product to market. This fact is what drove the formation of groups like TAPR to support the development of the mode in the first place.

However , my purpose here is not to enumerate EVERYTHING that is wrong, but rather to show how relatively easily ONE thing can be done right.

DCD is a "level one" issue, which mean8 that there is no standard document for TNC manufacturers to use as reference for guidance in this aspect of a TNC design. Writers of software for TNCs and other leve2 2 devices also have no guideline8 on what is reasonable to expect from the DCD circuit.

This is unfortunate* because proper DCD operation can make or break a packet channel. I have observed many thousands of unnecessary collisions and retries on VHF and HF packet channels (both amateur service and commercial as well) related to substandard DCD performance. After helping a number of hams drastically improve the operation of their TNCs here in the Tucson LAN, I decided to write this paper and share the results of my work with the Amateur packet community at large.

WHAT DCD SHOULD DO

The purpose of the DCD function in the modem used in a packet radio TNC is to prevent transmission on an occupied channel. If two stations transmit at the same time. we say a co22ision has occurred. This almost invariably means that both stations will have to resend the corrupted data. This has the effect of increasing the total load on the channel and reducing throughput for everyone sharing the channel.

Ideally. no station would ever step on another's transmission, and all stations would clearly hear all other stations (or no other station except the other one involved in the QSO!). The world is not ideal, however, so the best we can do is tune the protocol to minimize it8 sensitivity to this fact. This means considering at least some level 1 issues in the protocol specification.

We can also define how a proper DCD circuit should act and then see what we can do to implement such a circuit.

There are four key feature8 that a DCD circuit should possess:

First. the DCD circuit must be able to reliably distinguish a data carrier from noise or other non-packet signals. An open squelch circuit, for example. should not inhibit transmission of packet data. There is no point in not transmitting because of receiver noise in the absence of signals. A DCD circuit which require8 the operation of the TNC with audio gated by the typical narrowband FM radio squelch circuit, directly contributes to increased co21ision frequency. It does this by increasing the "deaf time" of the packet radio system. The TNC has no way of determining that another station ha8 keyed up if this happens to occur during the system "deaf time". The total system deaf time (neglecting the presence of a duplex repeater or regenerator for now) is the sum of:

1. The radio'8 transmitter keyup delay. That is, the time that elapses between assertion of push-to-talk (PTT) and the appearance of an on-frequency- full power, correctly modulated packet radio signal on the channel. Transmitters commonly used in amateur packet radio service exhibit de2ays ranging from approximately 60 to over 500 milliseconds for this parameter. Most commercially available synthesized radios are in the 150 to 200 millisecond range.

2. The radio's squelch circuit attack time. This is the time between the appearance of a signal just above the squelch threshold at the receiver input and the appearance of usable audio at the modem input. Radios commonly used in amateur packet radio service exhibit delays ranging from approximately 80 to we21 over 750 milliseconds (no that is not a misprint!) for this parameter. Most commercially available VHF FM radios currently available fall in the 150 to 300 millisecond range.

3. The modem's DCD attack time. This is the time between the appearance at the minimum usab2e signal-to-noise ratio (SNR) of decodable packet audio at the input of the demodulator and the assertion of the DCD output. Typical va2ues for this parameter range from approximately 8 to a little more than 30 milliseconds. These numbers are for 1200 baud modems. Phase information DCD circuits which use the recovered data stream to make the DCD determination will have delays which are proportional to the baud rate. Five character period8 is typically 2ong enough to produce a DCD circuit with a zero false detection rate when monitoring uncorrelated noise. This is a little over 30 milliseconds at 1200 baud.

AS you can see, even the worst case DCD circuit is better than the best of the squelch circuits. If we take the middle of the typical time8 for the various delays we see that eliminating the squelch delay reduces the "deaf" time from approximately 420 milliseconds to approximately 195 milliseconds. Doesn't seem like much? Then consider that this will reduce the collision frequency by approximately 50 percent on a channel with no hidden terminals. Depending on the DWAIT, FRACK, and RESPTIME settings of the TNCs using the channel, and the amount of traffic on the channel, this can and usually doe8 have a dramatic effect on the the channel throughput;.

Second, once a data carrier decision has been correctly made, it is important that the DCD indication remain valid through short fades, collisions, and periods when a signal too marginal to decode is on the channel. This prevent8 a TNC which is holding off its transmission from transmitting over a station which has a marginal signal, or beginning to transmit over *a* station which is still transmitting but whose signal received a short multipath hit during the packet. The DCD "hang" time also prevents the phase information based DCD circuits from "piling on" 8 collision between two other stations on the channel. The appropriate amount of "hang" time will depend on the propagation characteristics Of the band being used and the exact type of modem (if other than straight tuo tone FSK or AFSK). I have been having good success at 300 baud on HF FSK and at 1200 baud on VHF FM AFSK with a hang time of approximately 8 character periods.

Third, it is important that the DCD system NOT be sensitive to audio amplitude variations. It ehould respond in exactly the same way for any signal that the modem is capable Of decoding regardless of absolute input amplitude. This will prevent transmission over a relatively weak station who has keyed up immediately after a much stronger signal has terminated. For this to be fully effective the modem must have a large dynamic range.

Fourth. a8 can be seen in the above discussion, the modem should not take an unreasonably long time to assert the DCD output signal. An attack time of 5 character period8 seems to me to be a reasonable MAXIMUM allowed DCD delay specification.

EXISTING IMPLEMENTATIONS

There are tuo primary method8 of determining the presence of a data carrier in use in most TNCs today. These are phase information based DCD and amplitude information based DCD.

Phase information based DCD circuit8 look for coherent (phase related) information in the audio presented to the demodulator or in the data stream emerging from the demodulator. The TAPR TNCs use the in-phase carrier detector of the phase locked loop (PLL) in an XR2211 demodulator to look for phase information in the incoming audio. Thie type of detector directly detects the presence of the data carrier.

It is also possible to design a system which uses information derived from baud period phase relationships of the demodulated data stream to infer the presence of a data carrier. A good example of the use of this type of circuit is the KSNG 9600 bps modem. [1]

Either of these two phase information based methods are demonstrably superior for use in the radio environment than any of the amplitude information based methods with uhich I am familiar.

Amplitude information based circuit8 simply look for energy in the modem passband. Any signal is assumed to be a data signal. These circuit8 are appropriate only for telephone systems uhich are generally very quiet in the absence of the desired signal. Telephone modems are typically not required to operate in a Carrier Sense Multiple Access (CSMA) environment 80 their DCD requirements are much less stringent than demanded by our radio environment. Popular single chip modem8 are usually of this type, including the AMD7910 and TI TCM3105 chips.

Apart from the MFJ-1278, uhich incorporates the XR2211 improvement8 shown later in this paper, I am not aware of any Amateur packet TNCs uhich have modem8 which are fully optimized for the amateur packet radio environment.

IMPROVING EXISTING DCD CIRCUITS

Much of the following material is either taken from or is based on information found in articles I have previously published in several low circulation newsletters. Some of it had been simply sent as messages in response to questions on the linked packet bulletin board system. Don't be surprised if some of this material look8 familiar to you. This is the first time this ha8 all appeared in one place. Some recently discovered errors are corrected in this version. [2],[3],[4],[5]

The information presented here will allow the owner of almost any existing TNC to upgrade the performance of the DCD circuit in hi8 TNCs modem for proper operation on a packet radio CSHA channel. The vast majority of TNCs currently in service are covered. Only units which do not use the popular Exar PLL modem chip set or uhich don't have an appropriate baud clock signal available for use by an external synchronous modem are not addressed. These unit8 are only a tiny minority of the TNCs currently in use.

It is not particularly difficult to make a DCD circuit uhich operates in accordance with the points mentioned above. And, making the changes in your station TNC(s) will make a noticeable improvement in operation. At least, it has in the Tucson LAN where many TNCs have been modified for proper DCD circuit operation!

I should mention that the Tucson area is using a duplex audio repeater to eliminate hidden terminals from the system. A duplex regenerator would have the same effect if properly implemented. I believe that eventually it will be seen by everyone involved that all user access to netuork nodes should be by uay of either repeater or regenerator.

XR2211 IMPLEMENTATIONS

Units such as the TAPR TNC 1 and TNC 2 can be easily modified to have dramatically improved DCD performance. Figure 1 is a

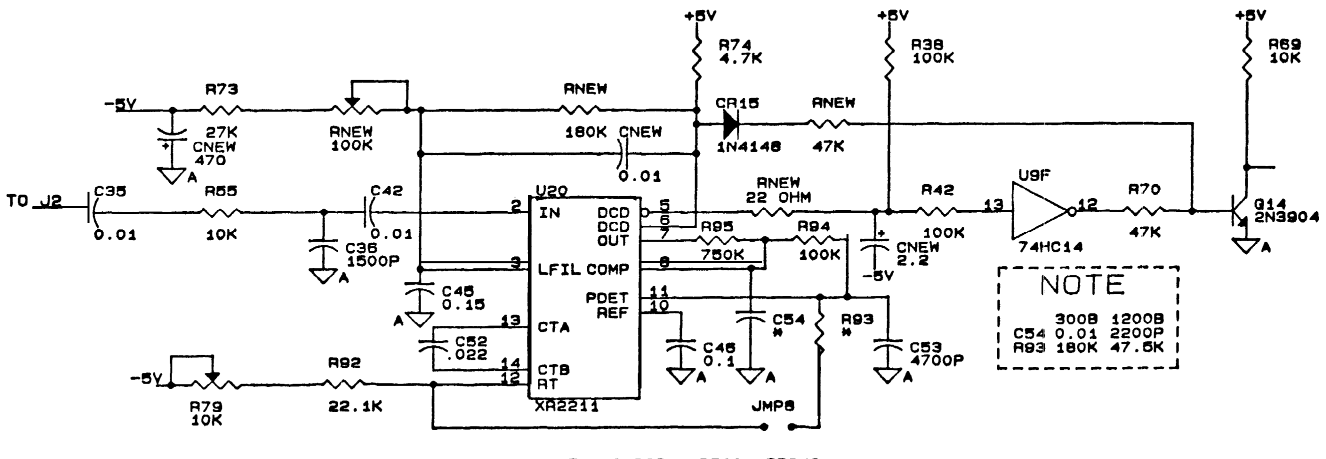


FIGURE 1 - TNC 2 DCD MODIFICATIONS

circuit diagram of a 2211-based demodulator which has the characteristics noted above.

Hysteresis is employed in the DCD decision threshold and "hang" time is added by the 74HC14 circuit following the modem chip.

I have incorporated this circuit in many TNCs and literally hundreds of commercial packet communications controllers. The improvement is very noticeable. Check the schematic diagram of your TNC. You will probably notice different value8 used in the various feedback resistors and capacitors used. That's because most TNC's 2211 demodulators are based on Exar Application8 Notes or Data sheet information which assume8 that the chip is to be used to implement a land line modem. The change8 presented here have been incorporated in some commercial packet systems designed from the ground up for radio appl ication. The improvement in performance really is noticeable. Try it!

The following modification procedure gives a step by step process to follow for modification of the modem in the TNC-2. The same modifications can be performed on a TNC-1 or beta teat TNC-1 board but the part numbers given in the procedure will not apply. Also, it will be necessary to patch a 74HC14 into the wire urap area of the board to construct the "hang time" generator.

If the complete modification including the variable threshold control is done on a TNC-1 or Beta Board. . extreme caution should be exercised to assure that the threshold control is not set to defeat the DCD entirely. Software available for these TNCs ha8 no provision for detecting a DCD fault condition. For this reason it is recommended that if this modification is done to a TNC-1 or Beta Board, you should include hardware gating of the data output from the modem based on the DCD output signal. This will prevent reception when the DCD circuit is disabled and thereby warn the operator that something is wrong.

MODIFICATIONS TO TNC-2 DATA CARR I ER DETECTOR (DCD) CIRCUIT

These modification8 are to allow correct DCD operation in a TNC-2 modem, Variable DCD decision threshold, is included to al low compensation for various audio bandwidths presented to the demodulator when using appropriately narrow filter8 on HF.

NOTE! For 1200 baud HF linear mode FSK work on 10 meters, the normal 2.4 KHZ SSB filter constitutes a “narrow” f i l ter for the demodulator.

There are three objectives to these modifications:

1. Provide threshold control for the DCD circuit. This allow8 the operator to adjust the demodulator to compensate for the DCD threshold shift which accompanies bandwidth limiting the audio fed to the demodulator when a narrow filter is used in the radio.

2. Add hysteresis to the Data Carrier Detector. This reduce8 the DCD's sensitivity to noise. It doe8 this WITHOUT DEGRADING DCD ACQUISITION TIME for a valid data carrier.

3. Add DCD hang time on release.

The modification8 presented in Appendix A will upgrade the modem in any TNC-2 or clone (including MFJ-1274) which has a 2211 demodulator to the level of DCD performance of the 1278 modem.

OTHER TNCS

Other TNCs may require somewhat more drastic measures. If you have a TNC which uses either the AMD7910 or the TCM3 105 single chip modem. or a TNC which uses a modem based on audio filter8 like the PK-232. you can vastly improve the DCD performance of your modem for packet radio use . The circuit presented in Figure 2

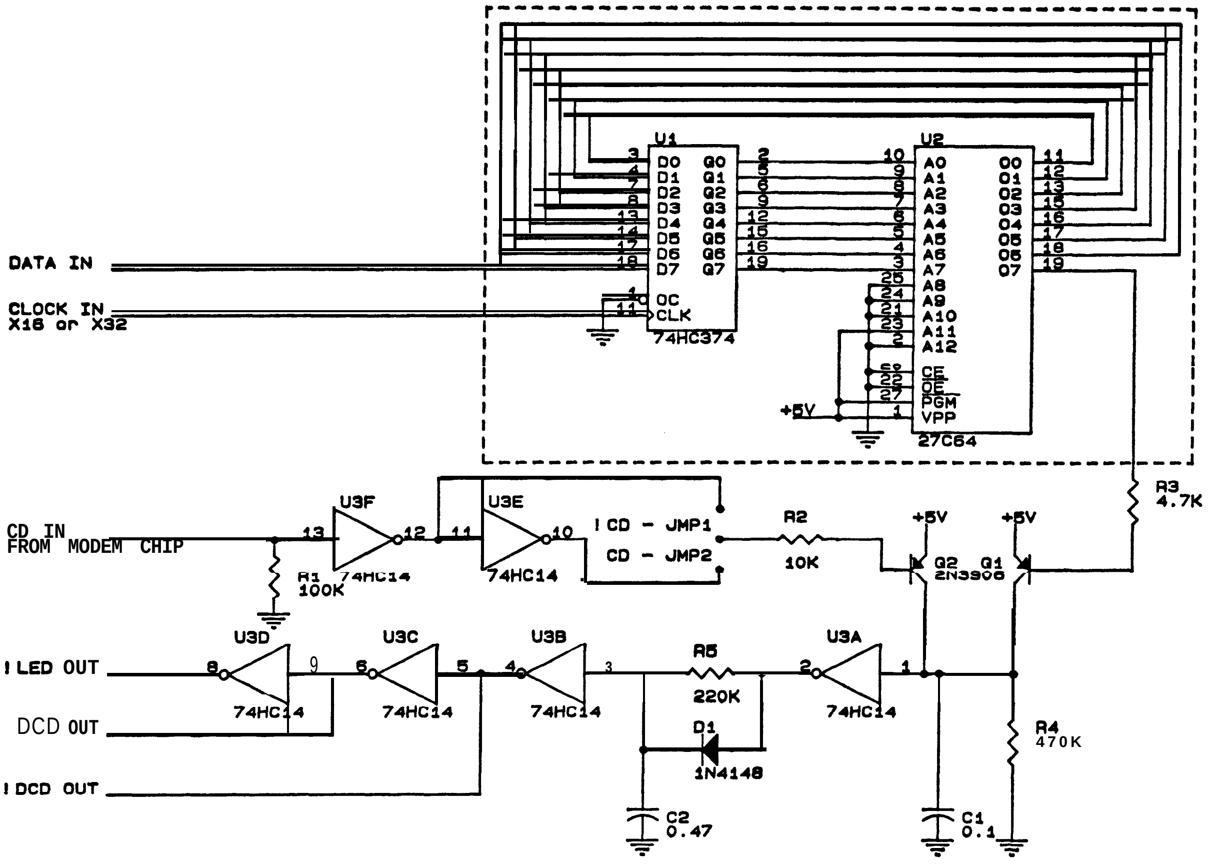


FIGURE 2 - IMPROVED DCD RETROFIT CIRCUIT

will provide a phase information based DCD for these TNCs which currently use amplitude information based DCD circuits.

If your TNC incorporates a TNC-2 style state machine (if it uses a Z80 SIO chip, it may; if it uses an 8530 SCC, it probably doesn't), you can add a phase information based based DCD by using the circuit shown in Figure 2 that is outside the dotted lines. If it doesn't incorporate such a state machine, you need the whole circuit as shown in Figure 2.

HOW IT WORKS

Please refer to Figure 2 for the following discussion.

The DCD circuit presented here is based on the update signals in a Digital Phase Locked Loop (DPLL) which recovers both baud clock and data from an NRZI packet data stream. Its output represents detection of baud clock phase coherence in the data stream.

The circuit consists of the state machine used in the TNC-2 and some delay elements used to make the DCD decision. The state machine is formed from the 74HC374 and the 27664 chips. The 74HC14 is used as a pair of retriggerable delay elements and for signal inversion and buffering.

The 27664 with the state machine code already burned into it can be obtained directly from TAPR. This same code is in the state machine ROM in any full TNC-2 clone which uses the 2211 demodulator and Z80 SIO.

One of the state machine signals (which was not used in the TNC-2) appears on pin 19 of the 27C64. This signal is the DPLL update pulse. As long as the DPLL is correctly locked to the incoming data, no pulses will appear on this pin. When the DPLL is not locked to an incoming data stream, there will be a continuous stream of pulses on this pin.

The DPLL update signal is used in this circuit to retrigger the first delay element so that it never times out so long as DPLL update pulses are present. If the pulses disappear, the delay element times out and generates the DCD signal.

The output from the first delay element keeps the second delay element triggered so long as DCD is true. When DCD goes false, the second delay element begins a time-out sequence which keeps the DCD output true until the time-out period expires. This is the source of the DCD "hang time".

While the circuit presented here is primarily intended for 1200 baud VHF FM operation. it will also work well for 300 baud HF packet work. If this is your application, the time constants on the delay elements will have to be increased.

The time constant of the "hang" generator (0.47 uF cap) will have to be increased for 300 baud operation to about 2 uF.

The time constant which is optimum for the DCD generator (the 0.1 uF cap in fig. 2) will depend on a number of factors including the bandwidth of the radio used ahead of the modem (You should use a 500 Hz IF filter in ANY radio used on 300 baud HF packet regardless of the presence or absence of audio filtering in the modem!).

You should pick a value for the DCD generator delay capacitor so the DCD circuit produces approximately a 10 percent duty cycle of false DCD "ON" time while monitoring receiver noise on a channel which is ABSOLUTELY free of ANY signals which fall within the demodulator's passband. The DCD generator delay capacitor will probably need to be somewhere in the range of 2 to 4 times the 0.1 uF value used for 1200 baud.

Both negative true and positive true DCD outputs are provided so that you may use the polarity which is required by your TNC. Also. JMP1 and JMP2 allow the DCD circuit to be configured to operate correctly from either a positive or negative true CD output from whichever modem chip is found in your TNC.

TNC SIGNALS

Once you have constructed the DCD circuit. you will have to obtain some signals from your TNC for the new DCD circuit to use. You will also have to arrange for the output of this circuit to be substituted for the normal DCD signal used in the TNC.

The signals required for the DCD circuit operation are:

1. A sample of the data recovered by the demodulator in the modem.

2. A sample of a clock which has a frequency of either 16 or 32 times the baud rate (X16 or X32 baud clock).

3. The intercepted Carrier Detect (CD) signal from the modem. This is the CD generated by the modem based on amplitude of the input audio.

4. A source of + 5 volts. If you use all CMOS parts. the power supply current requirements are minimal. The 74HC14 MUST be a CMOS part for the circuit to work properly.

5. Ground

If your TNC has provision for a TAPR style modem disconnect header, these signals (including the X16 or X32 baud clock) will be easily located and conveniently interfaced at this header. If it doesn't have this header. you will have to fish around in the circuit of your TNC on your own to locate them.

In any case. the DCD signal currently used in your TNC will have to be disconnected and rerouted through the new circuit.

STANDARD HEADER SIGNALS

The signal locations On the TAPR standard modem disconnect header are as follows:

1. Receive Data is obtained from header pin 18.

2. Carrier Detect is obtained from header pin 2.

3. Data Carrier Detect (DCD) is inserted at header pin 1. Jumper from header pin I to header pin 2 is removed.

4. The baud clock is obtained from header pin 12. The frequency of this clock will be either 32 times the baud rate or 16 times the baud rate depending on whether you have a TNC-1 or one of two types of TNC-2. No changes are necessary to make use of either clock speed.

AR7910 CONNECTIONS

The signals of interest on the AMD7910 modem chip are:

1. Receive Data output (RD)-----> pin 24

2. Carrier Detect (CD)----------> pin 25
   This signal is negative true for the 7910 chip.

TCM3105 CONNECTIONS

The signals of interest on the TCM3 105 modem chip are:

1. Receive Data output (RXD)----> pin 8

2. Carrier Detect (CDT)---------> pin 3
   This signal is positive true for the 3105 chip.

3. In TNCs which use the TCM3105 chip but do not provide another source of the baud clock, like the Kantronics KAM, you can use the signal at pin 2 of this chip. This signal. is very close to 16 times the baud rate (19.11 KHz instead of 19.2 KHz for 1200 baud).

COMMERCIAL TNC SIGNAL LOCATIONS

The information you need to find the proper signals in several commercially available TNCs is presented in Appendix B. This is not intended to be a complete list by any means. It is simply a list of TNCs which have been successfully modified to include this circuit.

CONCLUSION

This paper outlines desirable characteristics in a TNC's DCD circuit. Modification instructions have been presented to enable owners of existing TNCs to upgrade their units. Hopefully, TNC manufacturers will take time to investigate their TNC DCD implementations and make the minor changes necessary to reduce unneeded retransmissions on our crowded packet frequencies!

[1] Goode. Steve, K9NG, "Modifying the Hamtronics FM-5 for 9600 bps Packet Operation? ARRL Amateur Radio Fourth Computer Networking Conference. pp.45-51.

[2] Gustafson, Eric. N7CL, "HF Modem Performance? Packet Radio Magazine, DEC 1986, p.12.

[3] Gustafson. Eric. N7CL, "HF Modem Performance Comparisons? Packet Radio Magazine, JAN-FEB 1987. pp.21024.

[4] Gustafson, Eric. N7CL, "Letters To The Editor", Packet Radio Magazine- APR 1987. pp.8, 9. 19.

[5] Gustafson, Eric, N7CL, "Improved Data Carrier Detector (DCD) for 2211". RMPRA>PACKET, AUG 1988. pp.29,30.

APPENDIX A

MODIFICATIONS TO TNC-2 DATA CARRIER DETECTOR (DCD) CIRCUIT

Part numbers referred to are for the original TNC-2 as produced (briefly) by TAPR. I have not reviewed schematics of clones produced by all manufacturers so I cannot be sure that these numbers will be correct in all cases. Since I have seen information on the MFJ series. I can say that the numbers are ok for the 1270. 1270B, and 1274. If you are in doubt about part number correspondence. obtain a schematic of the original TNC-2 as produced by TAPR. Comparison between the TAPR schematic and the schematic of your TNC should resolve any differences. The TNC I used to verify this modification was a 1274 which I use primarily on HF packet. Since the physical layout of the modem area of the 1274 is different than the layout of the original TNC-2 or "pure" unaltered clones, I have avoided giving specific physical location information in the modification procedure.

Any manufacturer of the TNC-2 can feel free to incorporate this change into their hardware if they wish without incurring any obligation to myself or TAPR. I am available to answer any quetions in this area.

NOTE! Do NOT use TNC-2 firmware earlier than version 1.6 with this modification.

Firmware prior to V1.6 has no facility for detection of a DCD fault condition and therefore cannot warn you when the threshold control has been improperly set.

STEP BY STEP MODIFICATION

1. If you have a TNC-2 or clone (except MFJ-1274), and have not already removed the MF-10 filter and associated header parts, do so at this time. The reason for removing the MF-10 is that the operation of this filter circuit in the TNC-2 is marginal. This marginal condition drastically reduces the modem dynamic range. Simply remove both the MF-10 and the header associated with the MF-10. Then, under the board, solder a jumper between pins 1 and 8 of the header socket. Removing the MF-10 also unloads the negative 5 volt supply, improving its regulation and reducing the noise generated by the charge pump circuit. There is ABSOLUTELY NO PERFORMANCE PENALTY for removing this filter. The same filter used in the TNC-1 is NOT marginal and there is NO reason to remove the MF-10 from a TNC-1 or beta board.
2. Replace C35 and C42 with 0.01 uF caps.
3. Remove the 470 K resistor at R73. Be careful not to damage the circuit board pads or traces as they will be needed later in the modification.
4. Remove CR13.
5. Replace R70 with a 47 K resistor.
6. Lift the cathode end of CR15 from the circuit board. Install a 47 K resistor in series with CR15. Solder one end of this resistor to the hole vacated by CR15's cathode end. Solder the other end of this resistor to the cathode end of CR15 above the circuit board.
7. Replace R74 with a 4.7 K resistor.
8. Form a parallel network consisting of a 180 K resistor and a 0.01 uF cap. Make this network as compact as possible as it will have to fit underneath the circuit board. Solder this network in place under the board. One end goes to pin 3 and the other to pin 6 of the 2211 socket (U20).
9. Replace R38 with a 100 K resistor.
10. Above the circuit board, using leads as short as possible. install a 470 uF, 10 volt rated electrolytic capacitor between the -5 volt pad on the tuning indicator connector header (J3, pin 1) and ground. Connect the

positive lead to ground. Remember that this cap will have to clear the cabinet so position it as near the board as possible.

11. Under the circuit board. solder a 2.2 uF, 16 volt rated electrolytic capacitor from the junction of R38 and R42 to the negative 5 volt supply. The negative end of this cap goes to the negative 5 volt supply.

12. Under the circuit board, solder one end of a 22 ohm resistor to the junction of R38 and R42. The other end goes to pin 5 of the 2211 (U20).

13. On the front panel. mount a miniature 100 K, linear taper potentiometer (yes, there really IS room for this). This will be the DCD threshold control. It will be used to set the DCD trigger point to the proper value.

    If linear mode operation is not contemplated. you can eliminate the 100 K potentiometer and the 27 K fixed series resistor referred to in the next two steps. Instead, install a fixed 180K resistor in place of R73. This is possible because there is not very much difference in the audio bandwidth presented to the demodulator from the various makes of NBFM radios.

14. Solder a wire from the wiper of the 100 K pot to the pad vacated by R73 which connects to C45 and pin 3 of the 2211 (U20).

15. Solder one end of a 27 K resistor into the pad vacated by R73 which connects to the negative 5 volt supply. Solder a wire from the other end of this resistor to one of the 2 remaining leads from the 100 K pot. Use the lead that is set to zero resistance when the shaft of the potentiometer is turned fully CCW.

When operating a TNC with a DCD threshold control, set the control so that the DCD LED on the front panel flashes occasionally when there is no signal present. The “false DCD” duty cycle should be approximately 10 percent.

When operating VHF FM with the radio squelched. the DCD will not false. If you MUST operate with the radio squelched (thus incurring the penalty of the additional delay time of the squelch circuit). set the threshold fully clockwise as described below.

The audio bandwidth of some VHF FM radios is so wide that the DCD will not false regardless of the threshold control setting. This will almost always be true when the audio is obtained ahead of the radio’s squelch controlled stage before de-emphasis. For these radios simply turn the control fully clockwise. This sets the DCD to maximum sensitivity. DCD operation will not be impaired.

This completes the TNC-2 modem modification.

APPENDIX B

COMMERCIAL TNC SIGNAL LOCATIONS

AEA PK-87

It is relatively easy to interface this new DCD circuit to the PK-87 because there is no requirement to switch back to the internal DCD circuit once the modification is installed.

The Receive data signal is obtained from the center pin of JP4.

The Carrier Detect signal is obtained from the end of JP5 which connects to the modem chip.

The DCD output signal from the new circuit is inserted at the center pin of JP5. Use the NEGATIVE TRUE output, The jumper originally installed at JP5 is removed. The DCD indicator on the front panel will show the action of the new DCD circuit.

The X32 baud clock signal is obtained from pin 13 of U20 (a 74LS393 divider)- Don't be tempted to get this signal from the “clock” line on J4, the external modem connector9 as this is a Xl clock.

AEA PK-232

The PK-232 is also relatively easy to interface.

The Receive Data signal is obtained from the center pin of JP4.

The Carrier Detect signal is obtained from the end of JP6 which is NOT connected to pin 3 of the external modem connector.

The X32 baud clock signal is obtained from pin 13 of U8 (also a 74LS393 divider).

The DCD output from the new circuit is inserted at the center pin of JP6. Use the NEGATIVE TRUE output. The jumper originally installed at JP6 is removed.

To use the new DCD circuit with a PK-232 on VHF FM 1200 baud:

1. Set the audio level from the radio so that the tuning indicator "spreads" fully on the station with the lowest transmitted audio level on the channel.

2. The existing DCD threshold control should be set so that the existing DCD indicator LED on the front panel light8 up whenever there is ANY eignal or noiae input to the TNC from the radio.

Be sure that even the etation with the lowest amount of audio on the channel lights this LED. This LED should extinguish when there is no audio input from the radio (dead carrier from repeater etc.).

If you wish to observe the action of the DCD signal generated by the new circuit, add a high efficiency LED and lk series resistor between +5 volts and the LED output of the new DCD circuit. The anode end of the LED should go towards +5 volts.

Pat-Comm TINY-2

The Pat-Comm TINY-2 hooks up as follows:

The Xl6 baud clock signal is obtained from U1O pin 1.

Receive Data is obtained from J5 pin 17.

Negative true Carrier Detect (CDT) is obtained from 35 pin 2.

NOTE: This is an inverted version of the CD output from the TCM3105 chip itself. Since this is a negative true logic signal, JMP1 on the new DCD circuit will be used instead of JMP2 which would normally be used for a TCM3105.

Negative true DCD from the new circuit is applied to the TNC at J5 pin 1. Remove the connection between J5 pins 2 and 1. The existing DCD indicator LED will not show the action of the new circuit.

If you wiah to observe the action of the DCD signal generated by the neu circuit, add a high efficiency LED and lk series resistor between +5 volts and the LED output of the new DCD circuit. The anode end of the LED should go towards +5 volts.

If you wish to observe the action of the new DCD circuit on the existing LED indicator, you will have to do the interface a bit differently. First, you will get the negative true CDT signal from pin 1 of JPD. Then insert the LED output signal from the new circuit at either pin 2 of JPD or pin 2 of J5. Remove the jumper currently installed at JPD on the TINY-2 circuit board. If the new circuit is interfaced in this manner, the "RFDCD" signal can no longer be used. (This is no great loss, however, as it will also no longer be necessary.)

Kantronics KAM

For 1200 baud operation the signal location points of interest in the KAM are as follows:

The Receive Data (RXD) signal is obtained from pin 8 of the TCM3105 modem chip.

The Xl6 baud clock signal is obtained from pin 2 of the TCM3105.

The POSITIVE TRUE Carrier Detect (CDT) signal from the modem is obtained from pin 3 of the TCM3105. This line from the modem to the CPU is labeled with two numbered pads (7 and 8). These numbered pads represent pin numbers on a 20 pin modem disconnect header which is physically similar but electrically dissimilar to the atandard TAPR modem disconnect header. The connection between these 2 locations should be broken. JMP2 on the neu DCD circuit will be used.

The DCD output from the new circuit is injected at pin 21 of the 63B03 CPU.

The front panel LED which normally indicate8 the CDT signal activity will show the action of the new DCD circuit.

Finger - A User Information Lookup Service

Michael T. Horne, KA7AXD

19595 SW Martin St
Aloha, OR 97007
503-591-0488

ABSTRACT

With the recent explosion in amateur TCP/IP activity, primarily made possible by the KA9Q Internet Package, the need has arisen for a user information lookup service. Users on the amateur network can now retrieve important information about other amateurs through the use of a new application called finger. This paper describes finger and its potential as an important source for information retrieval in the amateur networking world.

1. Background

With the release of the KA9Q TCP/IP Internet Package, written primarily by Phil Karn with contributions from others, amateurs have been able to build computer networks based around their own personal computers. The package offers the amateur a complete system for performing file transfers, sending and receiving mail, remote computer log-in facilities, and simple keyboard-to-keyboard conversations, far more advanced and powerful than a standard Terminal Node Controller. What has been lacking is the means for a simple user information lookup service, allowing amateurs to exchange basic, yet important, information about each other. In the commercial world of networking and UNIX computers, such an information service exists. This service is called *finger.*

Finger has its origins at the University of California at Berkeley, written as a means for users of the UNIX system to retrieve information about other UNIX users [1]. A user on the system could get information such as the user's full name, his telephone number, what project he was working on, and other useful facts. Other system dependent 'information, such as whether or not he was logged on and what terminal he was using, was returned to the person requesting the information. Not all of this information is directly applicable to the amateur networking world, but something similar to this could be very useful.

2. Finger for the Amateur World

Amateurs have historically been curious about their fellow hobbyists. Most conversations you listen to on the HF bands consist of an exchange of name, location, and other personal information. Likewise, amateurs using packet radio query each other for this same information in order to get to know each other. Up until now, one has had to ask the amateur personally for the information, or lookup the information in a Callbook. Seeing the need for some sort of automatic information lookup service, I proceeded to write an application for the KA9Q package that would allow amateurs to lookup information when they need it.

The finger application I wrote allows amateurs using the KA9Q package, hereafter called 'net', to retrieve and provide information about themselves. Users can now retrieve such important facts as name, address, and telephone number, QSL information, station equipment used, projects currently undertaken, and many other things. In fact, there is virtually no limit to what information can be exchanged! **At** the same time, amateurs have complete control over what information about themselves can be retrieved by other hams. As our network expands, this application will help hams find out information about each other quickly and efficiently.

3. How to Use Finger

The finger command under net can be issued in any of the following three ways:

1) finger user
2) finger user@host
3) finger @host

User is the user's name you wish to query and **host** is the name of the host, or computer, that the user is at.

The first form of the command is used to find out information about a user at the local host, namely your own system. It is useful for testing finger on a system that you know is running. The second form of the command is used to find out information about a user at a remote host. If you don't know the name of a particular user at a remote host, you can use the third form of the command. This command returns a list of all users currently known on the remote computer.

To enable the finger server on your system so that others may query the users on your system, you must type 'start finger'. If you don't start the finger server on your host, other systems will not be able to finger users on your system.

4. The Finger Information Files

By now you are probably wondering how the computer knows information about a particular user. In order to provide a certain amount of privacy, *finger will only return information that the user* ***provides*** *in a* file. Each user maintains a text file of information. The local user's text file is returned to the remote user each time the local user is 'fingered'. If a user does not want to have information about himself sent to other systems trying to finger him, he simply does not create the file.

On an MS-DOS system running net, all of the finger files are stored in directory **\finger.** Each user wishing to be recognized by the finger system must create a **user.txt** file in the finger directory. For example, on KA7AYF's system he may have two users; 'glen' and 'lisa'. In order for net to recognize these two users, he must create two text files: \finger\glen.txt and \finger\lisa.txt .

What you put in the finger information files is completely up to you, but here are some guidelines:

1) You will probably want at least your name, callsign, full address, and telephone number in your finger file so other hams can contact you.
2) You might add information such as your license class, station configuration, and occupation.
3) You may wish to add some information about what projects; you are currently working on. This will provide helpful information for others who have similar interests.
4) Remember that the longer your finger file is, the longer it may take to transfer the data to the system fingering you.

5. An Example Finger Session

On my system, I have two users; 'mike' and 'teresa'. If someone fingers mike@ka7axd.ampr, it might look something like this:

```
net> finger mike@ka7axd.ampr
SYN sent
Established
[ka7axd.ampr]

   Hello and welcome to ka7axd.ampr
    running the KA9Q TCP/IP code!


User:          mike (KA7AXD)
Real Name:     Michael T. Horne
Class:         Extra

Address:       Michael T. Horne
               19595 SW Martin
               Beaverton, OR 97007
               (503) 591 - 0488

System:        IBM AT Clone
               MFJ TNC2 KISS TNC
               Yaesu FT-27RB

Occupation:    Hardware/Software Engineer
               2710 Spectrum Analyzer Group
               Tektronix, Inc.

Close wait
Last ACK
Closed (Normal)
net>
```

6. Finger Internals

Finger uses the well known port number 79 using TCP. When fingering a remote host, a socket is opened to the remote host using port 79. Once established, the client sends the name of the user on the remote host to be queried, or simply a carriage return/line feed sequence if system information is desired (such as a list of known users). The server attempts to find a file in the finger database directory under the name user.txt. If it fails, it returns a short message to the client that no user with that name is known on the remote system, then closes the socket. If it succeeds, the server returns the contents of the user.txt file to the client, then closes the socket. If the form 'finger user' is used, a socket connection is attempted at the local host, and program flow follows that described above.

7. Summary

The finger application under the KA9Q Internet Package provides amateurs with a simple, but powerful, user information lookup service. Amateurs can use the application to quickly find important information about other amateurs on the network. As our network expands, finger's usefulness will expand with it, and may eventually serve as a primary resource for obtaining the information amateurs need.

Reference

1. "Finger(1) - User Information Lookup Program," *The UNIX Programmer's Manual,* University of California, Berkeley, 1986.

BIG SKY TELEGRAPH AND OTHER TALES

Dave Hughes WNKV696

Background

Let me begin by saying what I know about radio, including amateur, much less packet, you can put on the head of a pin. But thats where I started in computers 11 years ago too, so I guess it is no disgrace.

But I also know an important telecommunications development when I see it - as I did microcomputers in 1977 and modems in 1979. Thus, when Andy Freeborn NOCCZ, whom I had known for several years in Colorado Springs as we hacked out the mysteries of early Tandy computers, mentioned in 1982 something new called 'packet radio' I became keenly interested. My interest then, as now, was how to put grass roots communications technologies to general beneficial public use as I had been doing with dial-up computer systems since 1979 starting with a BBS and now with multi-user 386 unix with vpix, voice mail, fax, conferencing, email and data base use systems for local business, politics, and education.

I swiftly learned that either I had to go the Ham route, or buy pricey equipment from some of the big commercial radio companies. That there was no 'public packet' in between. When Motorola showed me a $4,000 device that cost more than an entire computer system for a small business, I knew that was not the route to take either.

I really admired the work TAPR had done in pioneering packet, but I knew that if I heeded the siren song of Ham Radio and became Ham-licenced just so I could use packet, that I would not be able to apply it directly to small businesses, politics or formal educational uses, or secure the traffic by encryptian - all the things that I was sucessfully developing using modem communications. So I decided to do it the hard way - get a business radio licence from the FCC that would permit packet - and assemble the system out of all the parts needed, at the lowest cost consistent with being usable by small business.

Pulling the Pieces Together

The long and the short of it was that, with no one I could find who had tried this route, it took several years to pull all the pieces together. And I had to work with some small local commercial radio companies who knew nothing about packet! I had bought a big fat copy of the Federal Telecommunications Regulations and I swear I read up on every frequency band which could conceivably be used for packet, and permit mobile operations with a base station. A group in the 150 mhz area looked promising. Then I learned one had to make application through some outfit called NABER which does 'frequency coordination* for the FCC. So a small commercial radio store monitored the local frequencies for several weeks before deciding that 157.62 was silent all the time in my area and application was made for that band. After an agonizingly long time I recieved a licence to operate at 157.62 mhz with a 100 watt base station in Old Colorado City and 6 mobile 50 watt stations. Which, all things being equal (and they never are) would let me cover all of urban Colorado Springs from a mobile, hopefully hand held, station.

One day, while all this was going on, an international trader business associate of mine - Larry Fox - showed up in Old Colorado City with two ministers of government of the Zulu nation of South Africa! They were interested in packet because their field operations have few telephones, but do have voice radios.

Partly because Gwyn Reedy of Pat-Corn operated a Florida non-packet radio bulletin-board (813-874-3078) I chose to do business with him - cause I could do so via modem. I started with his TNC-220. Then just as I was leaving for Montana to install and tend to a powerful small unix system that was designed to link the 116 one-room school houses of Montana to their teacher's college by modems, I got his newly released "Wireless Modem" - a TNC that was optimized for business use, had parallel printer ports, and even KISS - for eventual migration to TCP/IP and the Unix online world with which I was familiar.

I visited every radio store in Colorado Springs when I was looking for a pair of radios. Not one commercial one had a handle on packet, and the Ham stores knew a lot about packet but were not very helpful on business-frequency radios. Finally I settled on a pair of reasonably priced radios from Neutec - a 45 watt transceiver which could be used as either a base station, or operating at 12 volts, be mobile in a vehicle, and a 5 watt hand-held Neutec 'Marathon' which had the critical feature of built in external speaker and microphone ports for attachment to the controller.

Then on a local computer bulletin-board a helpful ham said that since 157.62 was so close to the ham 2 meter band frequencies that he thought a Ringo Ranger-type antenna would be just dandy. That settled that. I got one.

Concept of Use

Now my concept of the value of packet radio to the general population is as much as an extension of telephone based modem communications as a thing seperate from it. For in situation after situation I saw that the value of using lap portable microcomputers and modems was always limited by 'where is the nearest phone?' Or worse 'Where is the nearest phone not-a-multi-line-pbx-and-with-an-RJll-jack.' Seldom found in any office. Even resorting to using external battery-powered modems which have provisions for acoustic cups at 1200 baud is only a partial solution. One is always hooked to the end of the phone line. And not all phones are accessible by RJ-11 plugs.

And for schools! Well I am a passionate advocate of the formal teaching of 'telecommunications' right along with computers in schools, starting at a young age. For we are as much in a global 'communications' revolution as we are a 'computer' revolution. Nowhere do I see the subject taught. Frankly I am even tired of recent college Computer Science graduates telling me they never were exposed to digital communications in all four years of college, and never heard a modem tone!

But whenever I have suggested to educators that they integrate telecommunications into their curricu1la , I get back the arguement that today's cost of telephone installation - especially into a modern large school whose architects never anticipated running phone lines to classroom areas, plus the running monthly costs of a phone dedicated to modem use, is a major deterrent. For large modern schools the problem is cost of installation through new walls. For tiny rural schools it is the monthly cost for an instrument which may be only used a few hours a week, and only 9 months a year or less.

So to me, using packet radio is the answer, with a packet and low power radio at the classroom computer, or even on a rolling tray with the computer, a packet and radio at the nearest already-installed phone in the building, or nearby building, with a second external modem properly configured to permit a 'patch' into the phone to give outside access! With a total capital investment for the entire setup of two packets and radios in the range of $600 to $1000, no real monthly running costs, the 'cost' problem for schools would be solved! (Or at least that 'excuse' dismissed conclusively)

Now I couldn't find any local hams who had used a packet hooked to a modem, rather than a computer. So I had to hack away with RS232 pin-outs, modem settings, and packet commands until I got that mighty modem AT command set 'OK' prompt back on my computer screen via packet and knew it was feasible.

Graphic Radio

I also wanted to do something else. I may not know my radio, but I really do know my modem communications. On one of my dial-up systems for several years we have supported dual Ascii-Naplps sessions. NAPLPS is a very important ANSI-Canadian Communication's Board standard for the encoding, transmission and display of animated color graphics and text between unalike computers over telecommunications. So important that the new national 'Prodigy' computer dial up advertising-marketting service by IBM and Sears, after spending hundreds of millions on it and 4 years - is based on Naplps. And new services are being brought out by Bell Canada.

No one yet has made a go of Naplps based services, but it is more than accidental that it is still being used as a standard. For it is highly compressed code - like 2 to 5 k for a full color screen, instead of 500k for a 'bit mapped' one. And it is terminal independent. What you see on an IBM PC you can see on a MacIntosh, a CAD/CAM workstation or a C-64. Thus, as all communications should be, it approaches universality.

But NAPLPS uses all 256 chars of an 8 bit set. It is a 'super-set' of ascii. Yet it includes Ascii. So what goes over the modem line is any possible combination of all 256 chars. Would packet handle this without corruption? Lets try, I said. So on a trip to the IEEE Conference in Monteray, California, my host happened to be Ham Marc Kaufman WB6ECE. He and his Ham son Matthew picked me up at the San Jose Airport and we drove down US 101. As any self-respecting Ham would have, Marc had his 2 meter gear in his van, his MFJ packet, and his son operated a Model 100 hooking up to a rich variety of packet stations, starting with a guy on a bike in Mountain View through digipeaters. Aha, I said. Can we try my computer? A Toshiba 1100 at the time. So with me issuing the Procomm and PC commands, he running the packet commands, we connected to a Red Cross Packet BBS, opened an upload file and uploaded without either error-checking from the terminal programs or the Transparent Mode of the packet a 5 k Naplps file. Then turned around and slipped a Naplps Terminal Program into the lap top, and called for the remote BBS to 'type' the file.

Voila! Uncorrupted, came back the code and smartly displayed the graphic of Old Colorado City's Roger's Bar on the screen.

We had Graphic Radio! Using a very compact, near universal graphic code standard! I knew we were home free.

Applications in Montana

So then I carried these just-proven-out (hardly perfected) techniques, economics, and standards to Montana, to introduce them to Western Montana College where, over the last six months I had been retained to set up a powerful desk top, multi-user unix dial in system to link the 116 one-room school houses of Montana in 'Rural Net' to their teacher's college. I designed the system running all the 300 features of any good unix system, including e-mail, computer-conferencing, data base with 6 phone lines, including 800 numbers and it went on line in February 1988. The system, called Big Sky Telegraph has been an instant success, with teachers who had never seen a modem, but getting one for their tiny school's Apple through the mail, logging on and for the first time connecting up with, not only the college, but each other across the state.

But as I suspected, in case after case, the *telephone* is not in the one-room school room. They have to lug the Apple to the phone. And school boards in schools with only 8 kids are understandably reluctant to go to the constant expense of an installed year round voice phone. So, my idea of a packet radio-phone patch was demonstrated to 9 faculty members of the college who immediately also saw the application as 'cordless lans' across campus to link valuable resources to each other without the cost of hard wiring, or the limitation of modem phone.

But there was another reason to introduce packet to the educational needs of this very rural state. To introduce adult non-traditional 'students' from the tiny communities to the newest, but low cost, communications technologies so that they might better introduce them into their daily work. No reason a struggling rancher who already has a microcomputer to perfect his herd shouldn't be able to track the fluctuating price of beef on the Chicago Board of Trade, or the markets of Billings, from a pick-up truck - or even a horse - while out on the range. And enter orders to his agent.

Then the college itself, in Dillon, Montana is 60 miles from Butte, where the closest Tymnet Packet Switching phone node is. Since part of the idea is to link those tiny schools, the teachers, students, and even ranchers, farmers, businesspersons from the surrounding community not only to the college, and each other, but the outside world via Unix's UUCP, Newsnets, international telecom networks, it will be necessary for Big Sky Telegraph to be linked to Tymnet. Either direct dial phone costs, or dedicated circuits are prohibitive for this purpose between Dillon and Butte. Packet radio digipeated off a tower, perhaps on Red Mountain to a local phone in Butte seem both economically and technically feasible,

Then as it happens, Asst Professor of Computer Education Frank Odasz, the key faculty member responsible for Big Sky Telegraph and his wife Reggie - both faculty members (with Master's Degrees in Educational Technology from University of Wyoming) live on a ranch 5 miles out of town. With one voice telephone. With two communicating computers. And the need to check into Big Sky in the early morning, evening and holidays. So the phone gets tied up too much. Could packet radio....?

So even before getting my packet rig set up in Colorado Springs, I carried it to Montana, and in a series of brief experiments, bench tests, demonstrations, and many calls to state officials who - as it turns out are using government packet radios for a variety of purposes- to determine both the regulatory and radio-technical scene, both very promising.

The attitude of those Montanan% toward both the modem and packet radio technologies couldn't be better. Already Big Sky Telegraph is famous across the state - with its integrated modem communications, voice mail, fax, optical scanner, naplps and over 7,000 messages having been left on it by exited teachers, student, faculty and outsiders who see the promise of digital devices overcoming their distance problems - which are both a blessing, for quality of life considerations, and a curse, for business and access to resources reasons,,

We had many laughs too, as the juices started flowing when the possibilites were discussed. Elaine Garrett, Assistant Sysop for Big Sky Telegraph is also a Fishing Guide and Outfitter. So we started sketching out what it would take to make the fly rod the antenna, put the electronics in the handle and reel, the battery in the tackle box and enable the big city guy to be a packet-connected stock-broker fisherman. And a three-day FCC licence to operate, just like a temporary fishing licence?

They want packet. I expect to be mosying up there in a few more months, help fill out their FCC licence applications, and as a consultant, system intergrator, and trainer, bring the first packet radio equipment to the small business and educational communities of Big Sky Country and integrate it into their already successful phone-modem based system.

The Importance of Ham Development

During the 6 years it has taken me to explore every alternative and finally put packet radio technology to practical and economical grass roots public use it became obvious that without the work of a large number of Hams over the past decade I could simply not have done it.

Oh sure, large communications or computer companies could have been doing research and development of this area. But when large corporations pay for their own development, they understandably try to keep the technology for themselves, or charge large fees for its use by others. It is clear that there would not be the rich and diverse development of packet devices and techniques - much of which has not even yet found its way into commercial systems - nor the really low cost access to the technology by 'we the people' without Ham Packet Radio development efforts - and FCC authority.

Potentially packet radio is to communications what micros are to computing. A technology for Everyman. It is going to take very wise Congressional and FCC policies to insure that it stays that way.

Packet to Go

Meanwhile here in Old Colorado City I have acquired an old brown brief--case in valise form, where I now carry not only my lap top portable Toshiba 1000, a 3 lb Diconix ink jet printer, wireless modem, power supply, and radio with telescoping antenna. 25 lbs. I now routinely log on to my base station on L57.62, go through an external modem to an outside line, and call any system needed through the patch. And never worry about finding a suitable phone.

So I am doing business by packet radio. And am working with Pat-Corn to deliver even a smaller packet controller with the radio built in. So others can do business too. At 17 lbs. Solar rechargable.

By daily use of packet in the small business-professional environment I am learning those things you who read this already know. But which the small business, small government, small educational world already needs to extend its micros and modem reach to every comer of America and the world.

I might even understand what I am doing with these radios one of these days.

INTERNATIONAL ROUTING DESIGNATORS

Lew Jenkins, N6VV
David B. Toth, M.D., VE3GYQ
H. N. "Hank" Oredson, WORLI

c/o Dr. D. B. Toth
499 Bobbybrook Drive
London, Ontario, Canada
N5X 1G8

It has become obvious by now that the work-horse of our so-called packet network is the venerable BBS program. In fact, some will argue that it has been too successful. Every time that a band-aid is needed to "fix" the network, it is applied through the various BBS programs. It is probably fair to say that the maintenance of the forwarding tables is a drudgery that most sysops could do without. This point also under-scores a serious problem faced by all networks: ROUTING.

With the introduction of WORLI V7.00 and support for Hierarchical routing designators, we have an opportunity to improve traffic routing particularly for international traffic. Since N6VV is at the present time responsible for traffic to Asia and the Pacific, and occasionally Europe and Africa, he has implemented some Hierarchical routing designators which will assist him in international routing.

Using this structure mail can now be addressed :

```
JA1ABC @ JA1KSO.JPN.AS
        or
VK4AHD @ AX4BBS.AUS.AU
```

Starting today you can begin using Continental and Country designators for international traffic destined for Asia and the Pacific. A forward file may be set up to support the following codes:

** Continental Designators **

NA	- North America
SA	- South America
EU	- Europe
AS	- Asia
AF	- Africa
AU	- Australia

** Country Designators **

For country codes there is a generally accepted international standard for abreviations. These are used in international electronic message standards such as ANSI X.12 and EDIFACT. They are published by the International Standards Organization and known formally as IS0 3166-1981(E/F).

Country codes (abbreviated list to show common country codes):

Argentina	ARG	Japan	JPN
Australia	AUS	Korea,North	PRK
Austria	AUT	Korea,South	KOR
Belgium	BEL	Lebanon	LBN
Bermuda	BMU	Liechtenstein	LIE
Bolivia	BOL	Luxembourg	LUX
Brazil	BRA	Malaysia	MYS
Brunei	BRN	Mexico	MEX
Bulgaria	BGR	Monaco	MCO
Canada	CAN	Morocco	MAR
Chile	CHL	Netherlands	NLD
China	CHN	New Zealand	NZL
Colombia	COL	Nicaragua	NIC
Costa Rica	CRI	Norway	NOR
Cuba	CUB	Pakistan	PAK
Denmark	DNK	Panama	PAN
Dominican Republic	DOM	Paraguay	PRY
Ecuador	ECU	Peru	PER
Egypt	EGY	Phillipines	PHL
El Salvador	SLV	Poland	POL
Finland	FIN	Portugal	PRT
France	FRA	Romania	ROM
French Polynesia	PYF	Saudi Arabia	SAU
German Demo. Rep.	DDR	Singapore	SGP
Germany, Federal Rep	DEU	South Africa	ZAF
Greece	GRC	Spain	ESP
Greenland	GRL	Sweden	SWE
Guatemala	GTM	Switzerland	CHE
Haiti	HTI	Syria	SYR
Honduras	HND	Taiwan	TWN
Hong Kong	HKG	Thailand	THA
Hungary	HUN	Turkey	TUR
Iceland	ISL	United Kingdom	GBR
India	IND	United States	USA
Indonesia	IDN	Uruguay	URY
Ireland	IRL	USSR	SUN
Israel	ISR	Venezuela	VEN
Italy	ITA	Yugoslavia	YUG

State and province codes shall be the recognized two-character code established by the American and Canadian Post Offices. These may also be found in the Callbook listings.

It is after we get down to the state/province/county level where the trouble may begin. To understand why, we must examine how the BBS code goes about matching things in the route. The first principle is that it attempts to find a match between the items in its forward file and the left-most item in the address field. As an example, say that we send something to WORLI @ WORLI.CA.USA.NA, and that the only entries that we have in the forward file are for CA. That match would be sufficient to allow the message to be forwarded. If WORLI were found, that entry would take precedence (because it is more left in the field than CA) and would of course also ensure delivery. The best way to look at it is"WORLI AT WORLI which is in CA which is in USA which is in NA". So far so good.

But the Japanese network wants to use area routing numbers. For example, JA1ABC @ JA1KSO.42.JPN.AS ... and everyone says, "So what,, let them!" Of course that is very mature of all of us, but the trouble is that the 42 in that string may also match wild-card ZIP codes that some folks keep in their forward file, such as 42*.The solution we propose is to use an agreed upon key character for designators below the state and province level, and we recommend the octothorpe, "#".

So now the above address would be JA1ABC @ JA1KSO.#42.JPN.AS . Other examples could be:

1) WORLI @ WORLI.#SFO.#NORCA.CA.USA.NA - WORLI within SFO (San Francisco) within North California, etc.

2) VE3BTZ @ VE3GYQ.#LONDN.#SONT.ON.CAN.NA - VE3BTZ at VE3GYQ in London, in Southern Ontario, in Ontario, etc.

There is another added benefit to this scheme. It involves Gatewaying between the BBS world and other networks, such as TCP/IP via SMTP. Much of the pioneer work in setting up the gatewaying protocols has been done by NN2Z, N3EUA, and PAOGRI, amongst others. The WORLI BBS package allows for the forwarding of mail between the BBS world and the SMTP world. Of note is the fact that the WA7MBL package has allowed such message exporting and importing for some time now. This means that we can take advantage of the the TCP/IP host-names and their domain or hierarchal format for forwarding. Thus it is possible to send mail from the BBS to VE3BTZ as ve3btz@pc.ve3btz.ampr.org or from SMTP to w0rli@w0rli.ca.usa.na and not have any ambiguity.

We expect that WA7MBL will also be implementing hierarchal routing in the near future. This system is still compatable with older style systems, as a system that handles hierarchal forwarding identifies with the H feature letter: [RLI-8.00-CH$]. If it does not get an appropriate response, it uses the left-most item in the "@ BBS" string as the "@ BBS" for the message.

The authors hope that this paper will serve as a starting place for improved message routing by means of implicit routing. Low-level (VHF) BBSs need only maintain state or province or country codes for distant BBSs, and route such traffic to their nearest HF Gateway. In turn, the HF station routes it to the desired state, where the receiving Gateway station would have a detailed list of the BBSs it serves.

Correspondence may be addressed to the address given at the start of this paper, or to VE3GYQ @ VE3GYQ.ON.CAN.NA or N6VV @ N6VV.CA.USA.NA .

AX.25 PACKET RADIO COMMUNICATIONS USING METEOR SCATTER PROPAGATION

Results from experiments performed during autumn 1987

by

Thomas Johansson, SM5IXE
AMSAT-SM

1. GENERAL

1.1 Introduction

Meteor scatter propagation is commonly used in modern commercial and military digital communications. This traffic preferably takes place in the frequency range 30 to 50 MHz, where the reflectivity and duration of the meteor trails are the best. Meteor scatter links exhibit very reliable communication, even during periods of low meteor trail occurrence [2].

Among radio amateurs digital meteor burst communication has been discussed for many years. Despite this, no experiments on amateur basis have (to the knowledge of the author) been performed in Sweden or elsewhere in the world. During the past two years several amateurs have reported the random appearance of AX.25 packets from very distant stations. The propagation of these packets can only be explained by meteor scatter reflections. This supports the idea to exploit these reflections for packet radio links.

The following paper reports on the realization and results of an initial experiment with packet radio communication using meteor scatter propagation in the 144 MHz amateur radio band. The tests were carried out during the meteor shower Geminides in the midst of December 1987.

1.1 Purpose of the experiment

The main purpose of the experiment was to explore the possibilities of using meteor scatter reflections for packet radio communication in the 144 MHz band. The investigation involved measurement of channel throughput and delay during a major meteor shower.

2. BASIC PRINCIPLES

2.1 Occurrence of meteor trails

Particles in the form of meteoric debris with masses ranging from only a few milligrams up to several grams occur, with roughly uniform density, in all parts of the world on a virtually continuous basis. Observational data and theoretical predictions, verified by recorded statistics, indicate that more than 10^{12} meteor trails of a size sufficient to support radio communication enter the atmosphere on a daily basis. ‘The incidence of usable meteor trails varies both diurnally and seasonally. Diurnal variations occur with a maximum about 06 AM and a minimum around 07 PM local time. The ratio of maximum to minimum occurrence ranges from 3: 1 to 4: 1. Seasonal variations exhibit similar characteristics in the Northern Hemisphere with a maximum in July and a minimum in February. Seasonal variations are opposite in the Southern Hemisphere [3].

Beside these seasonal variations in the mean meteor density, showers occur during short periods of time every year.

2.2 Trail ionization

Ionized meteor trails are created by the transfer of kinetic energy (related to the mass and speed of the meteor) from meteoric debris into the energy of ionization which leaves a trail of positively charged ions and free electrons. The free electrons form the medium for reflection or re-radiation of radio signals.

Two types of ionized trails are created. They ***are*** termed ***under-dense*** and ***overdense,*** and are typically categorized by the intensity of ionization in the trail. Underdense trails exhibit relatively small electron line densities in the range of 10 to 10^{14} electrons per meter. Underdense trails do not support sufficient ionization density to reflect radio signals. Instead, the individual electrons are excited by radio waves, and subsequently act as minute dipoles which re-radiate the signal. In meteor trails that exhibit electron line densities of greater than 10^{14} electrons per meter, a sufficient density of ionization exists to permit specular reflection of the incident radio signal.

These trails are termed overdense and the center part of the trail acts as a perfectly conducting cylinder.

Underdense trails are prevalent. However, overdense trails exhibit longer duration and lower attenuation than underdense trails.

2.3 Frequency dependence

Meteor scatter reflections vary strongly with operating frequency. Reflected or re-radiated signal amplitudes have been shown to be proportional to the inverse cube of the frequency ($1/f^3$) and its time duration is roughly proportional to the inverse square of the frequency (l/f?). The average duration of underdense trails is between 200 and 400 ms in the frequency range 30 to 50 MHz. Overdense trails may exhibit considerably longer duration - up to several tenths of seconds. With the above frequency considerations, the duration of underdense trails will be around 10 times shorter on 144 MHz than on 40 MHz, i.e. a mean duration of only 20 to 40 ms. This clearly demonstrates that underdense meteor trails can not be used for standard amateur packet radio traffic, since the packet length is roughly 0.8 seconds. The amplitude of signals on 144 MHz will be about 14 dB lower than corresponding signals on 40 MHz.

2.4 Maximum communication distance

The maximum range for a meteor scatter link is limited because of geometrical reasons. The reflection takes place at an altitude of 90 - 120 km and because of the earth's radius, the maximum communication distance is about 2 000 km.

3. EXPERIMENT SETUP

3.1 Selection of modulation form

The selection of a suitable type of modulation for the experiment system was done with great care. Signals propagated on meteor scatter paths are generally exhibiting small signal to noise ratios (SNR) and a doppler shift due to movements of the trail. Under these circumstances binary- phase-shift-keying (BPSK) was considered to be the most suitable modulation form. BPSK meets the need for low bit- error-rates (BER) at small SNR [4]. Furthermore, if the loop- bandwidth of the demodulator is correctly designed it is possible to accomplish a stable phase-lock on signals which differ from nominal frequency due to doppler effects. Finally, the hardware implementation of a BPSK-demodulator is straight forward.

3.2 Experimental hardware

Two identical experimental stations were constructed according to the following general specification:

Frequency band	144	MHz
Power output	150	W
Noise figure	<2	dB
Antenna gain	15	dBi
Antenna polarization	hor.	
Modulation	BPSK	
Bitrate	1200	bit/s
Protocol	AX.25 level 2 (ver.1)	

Heavy demands had to be put on transceiver frequency accuracy and long-term stability. ICOM-transceivers were tested prior to the test. IC-251E and IC-27l.E were finally chosen for the experiment system. The continuous transmit/receive switching puts heavy strain on the radio equipment, especially on the switching circuits in the linear amplifier and low noise pre-amplifier. To minimize this strain, a so called sequencer was constructed, which controlled the switching order of the transmitter and amplifiers.

Modems used in the experimental set-up were of G3RUH design [5]. This modem was originally designed for digital communication through the amateur radio satellite Fuji-OSCAR- 12. Only minor modifications had to be made for our needs.

The packet protocol AX.25 was handled by standard terminal node controllers (TNC), which had to be slightly modified to allow the connection of external modems.

3.3 Experimental software

The two stations were designed to be operated fully automatically. Hence it was necessary to write a computer program which controlled the link process and stored the results. The program was originally written in C-language in a UNIX environment, but was later translated to TurboBasic, running on IBM/XT. The master station was controlled by the computer. From this station all activities were initiated, according to a certain procedure; see below. The slave station was only running in “loopback-mode”, i.e. the TNC was provided with a loopback circuit in the terminal port, which allowed immediate retransmission of all correctly received packets.

A completed contact beween the two stations was realized according to the following procedure:

1. MASTER continously sends connect-frames <SABM>

2. SLAVE receives <SABM> and verifies by starting sending <UA>-frames

3. MASTER receives <UA>, stores the time on disk and starts the transfer of a standard file consisting of 400 bytes

4. When SLAVE receives an <I>-frame correctly, the frame will immediately be retransmitted, with updated "expected frame-number field"

5. When MASTER receives a correct <I>-frame, the next one will be transmitted

6. When all characters in the file have been transferred, the MASTER stores the time and starts sending <DISC>- frames

7. SLAVE receives <DISC> and sends <UA>

8. MASTER stores contact statistics, which are available from the TNC

9. The procedure is repeated

By this procedure a transfer of 400 bytes has been accomplished. Using the statistical functions supported by the TNC (DISPLAY HEALTH) is a convenient way to get detailed information on the link quality.

3.4 Time and station locations

The experiment was performed in the period 7 - 16 December 1987, i.e. during the meteor shower Geminides, which reached its maximum intensity on 12 December.

The master station (SK5BN) was located near Linkoping in the southern part of Sweden. The slave station (SK2GJ) was situated in Kiruna, above the Arctic circle which resulted in a great circle distance of 1100 km between the two stations.

5. RESULTS

On the second day of the tests, severe problems appeared caused by interference in the slave station receiver. The interference signals originated from two different sources. Reciprocal mixing between phase-noise in the receiver and the extremely strong signal from the VIE-beacon SK2VHG caused an increase in noise level by several dB, synchronuously with the keying of the beacon. Furthermore, man-made noise from the surrounding city contributed to the over-all noise level more than expected. The reciprocal mixing products were easily removed by a notch filter in the receiver input, but the over-all noise level was still too high to expect any success for the tests.

At this point, it was decided to change the experiment procedure, in that SK2GJ in Kiruna was to transmit continuously <SABM>-frames and SK5BN in Linkping was set to listen-only mode. All correctly received packets at SK5BN were stored on disk.

Furthermore, a mains power failure at SK5BN resulted in a loss of data after 13 December.

5.2 Extracted results

As an effect of the encountered problems with interference and power failure, the effective time :for the experiment lasted 10 December 4.30 PM to 13 December 8.30 AM local time. During this period totally 96 <SABM>-frames were correctly transferred beween the two stations.

It has not been possible to make any statistical conclusions from the very limited amount of data that was collected during the test. However, the results clearly show, the possibility of establishing 144 MHz packet radio links using meteor scatter propagation.

6. FURTHER EXPERIMENTS

In spite of the encountered problems, the results from the tests encourage towards further experiments. For instance, it would be interesting to establish a two-way-contact and to fully use the computer software that permits more statistical information to be collected.

The next experiment should suitably be assigned to the Persides in August 1988. If possible, the stations should be equipped with more output power, since 150 W is considered to be close to the margin. Also, extreme care should be taken to select proper test sites and to check the interference level in advance.

7. ACKNOWLEDGEMENTS

Finally, I would like to thank all people involved in the project team. Without their enthusiasm and skill, this experiment could not have been made. Working with a team ofdevoted people is a great opportunity which I appreciate immensely; and I am certainly looking forward to future projects within the group.

8. REFERENCES

[1]. Johansson T, Eriksson M, Using Meteor Scatter Propagation for AX.25 Packet Radio Communications - Project Proposal, Norrkoping 1987

[2]. Day W E, Meteor-burst communications bounce signals between remote sites, ***Electronics,*** December 1982

[3]. Richmond R L, Meteor Burst Communications, Part I: MBC Advances Assist C3 Objectives, ***Military Electronics/Countermeasures,*** August 1982

[4]. Oetting J D, A Comparision of Modulation Techniques for Digital Radio, ***IEEE Trans. on Communications,*** December 1979

[5]. Miller J, The **JAS-1/FO-12** Modem, Cambridge, October 1986

THE AMSAT/TAPR DSP 1 PROJECT: HARDWARE DESIGN

by

Lyle V. Johnson, WA7GXD

BACKGROUND

A couple of years ago, Tom Clark, W3IWI and Bob McGwier, N4HY, began involving a number of Amateurs in applying digital signal processing (DSP) techniques to Amateur radio.

Recently, AMSAT and TAPR have combined forces to jointly develop and distribute DSP hardware and software for the Amateur community. Tom and Bob have written fairly extensively about the potential applications of DSP in the Amateur environment.

The first hardware fruits of this marriage is called DSP 1. This paper is intended to provide a design overview of the DSP 1 hardware configuration.

HARDWARE OVERVIEW

The DSP 1 is designed as a set of PC boards that can be configured to individual requirements. At a minimum, the system consists of a metal cabinet' an I/O board, a DSP board, a power supply board and a loader board. In place of the loaders a general purpose processor (GPP) board can be installed for greatly enhanced operation and flexibility. For all but the simplest applications the GPP will be required.

This breakdown of function to PC boards has the advantage of allowing upgrades at reasonable cost, providing flexibility for adoption of future standards or refinements in interfaces, and simpler troubleshooting. It has the disadvantage of increasing initial costs Somewhat.

Cabinet

The system is designed around a Ten Tec B-series enclosure. This is an all metal, low profile box that can be shielded for RFI/EMI. The use of this enclosure increases costs over the type of case used by a TNC 2, but is less expensive than the "Gucci" cabinet used with the TNC 1.

I/O Board

The I/O board mounts on the back panel of the cabinet. All connections to the outside world are handled through this board. This allows internal interfaces to run at TTL levels for digital signals and -5V to +5V levels for analog signals. It also simplifies construction for the kit builder, eliminating wiring harnesses.

By standardizing all interface signal levels, future DSP and GPP boards may be developed to utilize improved technology without rendering DSP 1 obsolete. Future standards in radio and serial I/O interface levels can also be accommodated by replacing the I/O board rather than a far more expensive DSP or GPP board.

All connectors are well bypassed to ground for EMI/RFI. Radio interfacing (PTT, CW keying, microphone audio, speaker audio, up/down control, etc.) is handled here, along with level translation and filtering. Two radio ports are provided.

Serial I/O is handled at RS232 levels. Again, all level translation is handled on the I/O board.

A special TTL-level connector compatible with the modem disconnect arrangement used in TAPR TNCs and the TAPR PSK modem is also provided for those users who wish to use the DSP 1 as an external modem for an existing TAPR-compatible TNC.

Finally, power is routed into the DSP 1 through the I/O board for consistent filtering.

DSP Board

The initial DSP board is based on the Texas Instruments TMS320C15 processor. This is a first-generation DSP chip with some enhancements over the earlier TMS320 10. It is run at a clock speed of 25 MHz, and has a full 4k words of high-speed static RAM for program memory.

An AD7569 eight-bit analog I/O port made by Analog Devices is used, and the board has provision for two such ports. Eight bits provide for over 40 dB of dynamic range, more than most radios provide at their audio outputs. More bits are useful for test equipment, but significantly increase the cost of the device.

Anti-alias filtering is generally required for any analog to digital converter. This filtering is provided in the case of DSP 1 by a Gould S35.28 programmable low pass filter. This filter provides a 7th order elliptic function with with stopband attenuation beyond the dynamic range of the 8-bit A/D converter. Placing it8 frequency cutoff under direct control of the DSP chip provides maximum flexibility for applications programmers.

A multiple channel programmable timer (82C54) is mapped in the TMS320C15 I/O space and may be used to generate interrupts and/or control the sample rate of the A/D and D/A. A phase shifter. designed by WB6HHV, allows the DSP chip to lock onto the external signal, al lowing reduced sampling rates for a given signal after acquisition. This is a feature not normally found on DSP analog front ends. and should prove very useful in many applications.

Digital I/O is handled through CMOS eight-bit latches. This allows the DSP board to connect to a TNC and exchange TTL level data with the HDLC chip, thus acting as an external modem.

The 4k word memory has two access paths. An external device (loader or GPP) can suspend the DSP chip, access the 4k word area as an 8k byte area, and read or write to it. This is how the 32015 programs are loaded.

There is a series of single-bit latches provided to synchronize information exchange between the DSP and GPP boards for high-speed data transfers. Eight-bit latches are used to store the information bytes transferred in this manner.

Address mapping and general purpose logic functions are bundled into a reprogrammable logic device (Lattice GAL (r) or equivalent) to al low reconfiguration and reduction of parts count on this board.

CMOS parts are used exclusively for minimum power diesipation.

(A Look Ahead)

The next generation board (DSP 2) will have considerably more processing horsepower (Motorola DSP56001 or TMS320C25), memory and analog I/O resolution (12, 14 or 16 bits). It will probably contain FIFO buffers for all I/O. It will be designed to retrofit into the DSP 1 cabinet and utilize the GPP and other resources.

Power Supply Board

The power supply board provides regulated +5, -5, +10 to +12 and -10 to -12 volt8 from an unregulated +12 VDC nominal source. It utilizes switching techniques for efficiency, although the initial board may use simple charge pumps. There is nothing particularly unusual about this board.

Loader Board

The loader board is intended for low cost, modem-only applications. It allow8 the user to select from a number of programs stored in a single EPROM (4 programs in a 27C256, 8 in a 27C512). A press of a button halts the DSP board, l oads the selected program into the DSP memory, then restarts the DSP engine. This board uses only a few standard CMOS logic functions and will be very inexpensive (excluding memory!).

GPP Board

The GPP board allows the DSP 1 system to become a highly flexible and very powerful communications tool for the Amateur.

The GPP is based on the NEC V40 integrated processor and 72001 serial I/O chip. This allows software to be developed and debugged on a standard IBM PC (r) or compatible system, This board features two serial channels running full duplex under DMA, al lowing high speeds to be attained for use with such devices as the Heatherington 56 kbps modem.

Four (4) 32-pin bytewide sockets allow up to 1/2 megabyte of memory, with 64k bytes a typical minimum configuration. Battery-backed RAM and a CPU watchdog are included, along with a Centronics compatible parallel printer port.

CMOS logic and CMOS LSI are used exclusively in the GPP for reduced power consumption and high reliability. Programmable logic devices contribute to hardware design simplification.

Bytewide I/O ports to and from the GPP board allow high-speed data transfer with the DSP card. In addition, serial I/O supporting multiple protocols can be used to exchange data with the DSP section.

The net result is a truly flexible and powerful communications device. Narrow-shift RTTY (30 Hz shift for 60 WPM, for example), tracking filters, efficient HF packet, SSTV and FAX applications all become possible without compromised modem performance.

WHITHER SOFTWARE

The Dalanco-Spry Model 10 is being used by a number of Amateurs in the context of the overall DSP development project. This unit is a plug-in card for IBM PCs and compatibles. It allows development and testing of software for the TI 3201x processors. As such, it is an excellent development tool for the DSP 1.

There are a number of applications that have already been written and used by several Amateurs, including moonbounce work with OSCAR-class stations, weather satel 1 i te image decoders with software modems. PSK modulators and demodulators, etc.

In the case of DSP 1, the software is waiting on the hardware!

VRAP UP

The DSP 1 project represents a major effort by AMSAT and TAPR to provide the Amateur community with advanced* affordable technology to enhance digital communications. It provides a modular, flexible platform for experimentation and continuing development, in this field.

A8 the technology represented by DSP 1 becomes commonplace in the ham shack, the need for new hardware to use new communications modes should become significantly reduced. For example, in order to use FUJI/OSCAR 12 or one of the new MicroSat-based PACSATs, one present ly needs a PSK modem accessory for his TNC. Such a modem costs about $100 as a kit or $200 assembled. A DSP 1 arhould sell for less than the TNC and PSK modem combination. yet provide as much flexibility for the Same specific application. It will also do numerous other things well, such as a multimode digital controller (like the AEA PK-232 or Kantronics KAM or MFJ-1278), or a special-purpose communication8 device (multi-level SSTV or WEFAX o r weather satellite FAX decoder).

Digital signal processing techniques promise improved reliability in communications. The DSP 1 will help bring these techniques to Amateur radio.

MICROSAT PROJECT - FLIGHT CPU HARDWARE

by
Lyle V. Johnson, WA7GXD
and
Charles L. Green. N0ADI

ABSTRACT

AMSAT-NA is preparing a new class of satellite. Intended for low earth orbit (LEO), "MicroSat" will bring a new level of performance and flexibility to the satell i te user community.

MicroSat embodies bold advances in low-cost satellite engineering. A new approach to satellite flight computer design was required to attain mission objectives: low power, low cost, small volume and mass, high performance, high reliability. flexibility. significant mass data storage capability and high speed I/O.

This paper focuses on the design of the flight computer hardware and outlines some of its capabilities.

INTRODUCTION

The first planned launch of the MicroSat series is scheduled in early 1989 aboard an ESA Ariane-4 rocket. A total of four (4) MicroSats are scheduled to be aboard: two (2) uill be PACSAT missions [1], one (1) Will be a speech synthesizer for educational use (DOVE) and one (1) uill be a CCD camera experiment.

AMSAT, with support from TAPR, Weber State College (Utah), AMSAT-LU (Argentina) and BRAMSAT (Brazil), is designing and building these satellites on a fast-paced schedule. First proposed in late 1987. there will have been approximately one year elapsed time from concept to launch!

For the authors- it is reminiscent of the UoSAT/OSCAR-11 Digital Communications Experiment (DCE) schedule. The OSCAR 11 mission was proposed in late July. 1983. and del ivered to NASA for launch in January. 1984 (and launched March 1, 1984)! [2]

The PACSAT mission spacecraft will have five 2m uplink channels and one 70 cm downl ink channel. The modulation techniques will be fully compatible with FUJI/OSCAR-12 -- 1200 baud Manchester data fed to a standard 2m FM radio for the uplink and 1200 baud PSK data returned on the downl ink. Eight (8) megabytes of message storage will be available on each PACSAT. Other data rates and modulation methods may also be experimented uith via in-flight hardware reconfiguration.

GENERAL DESCRIPTION

A typical MicroSat occupies a cube about 9 inches on a side. The CPU module occupies a slice about 1-1/2 inches thick. The CPU power budget is only 1.5 watts.

The CPU module occupies two (2) PC boards and provides the following resources to the satellite:

Six (6) serial I/O ports capable of multiple protocols.

Six (6) channels of manchester decoding in the receive signal paths (five standard user and one special purpose).

One (1) serial I/O port dedicated to telemetry gathering and control of the various spacecraft modules. experiments and systems.

Six (6) DMA channels.

256k bytes of error detecting and correcting (EDAC) memory.

Two (2) megabytes of RAM organized as four (4) 512k byte banks with dual-port access. .

Eight (8) megabytes of RAM organized as a serial-access pseudo-disk.

One eight-bit A/D for analog signal measurement.

Reference voltage for telemetry systems throughout the spacecraft.

Eight (8) I/O ports mapped for external experiment support.

Watchdog timer to automatically reset the CPU upon detection of a CPU crash.

Ground control reset capability in case all else fails.

SPECIAL FEATURES

The CPU module utilizes CMOS logic and CMOS LSI chips throughout. It employ8 a true multitasking kernel and takes advantage of a reduced power wait state when the task scheduler al lows. To reduce volume requirmeents, surface mount technology (SMT) ICs are used where available.

The microprocessor selected is the NEC 70208 (V40) CMOS integrated device. This IC contains a superaet of the Intel 8088 microprocessor (including added instructions and reduced cycle times for a given clock speed) along with several peripheral functions The on-chip peripherals include: clock generator: four (4) channel DMA controller (71071-compatible); UART (8251-compatible subset) ; three (3) sixteen-bit programmable timer/counters (8254-compatible); eight (8) input priority interrupt controller (8259-compatible); programmable wait-state generator and bus interface logic/drivers.

Additional DMA support is provided by a NEC 71071. a high-performance CMOS four (4) channel DMA controller. In conj unction with the V40 CPU. and allowing for the various modes and features used in the system. a total of six (6) ueab 1 e DMA channels result.

Serial I/O (72001) chips are also from NEC. These devices are similar to the Zi log Z85C30 SCCs used in some Amateur packet applications, but provide a number of interesting benefits. The most important ones are (a) ability to interface to the V40 and 71071 with a reasonable amount of discrete logic glue and (b) independent programmable baud rate generators for each transmit and receive channel.

A Harris 2k byte CMOS fusible-link PROM is used for bootetrap memory. A smaller version of this same device was successfully employed for the same purpose in the UoSAT DCE.

EDAC

The EDAC memory posed an interesting challenge. This type of memory is necessary to store program code. vectors and certain data that cannot be al lowed to become corrupted due to radiation effects.

The UoSAT DCE has 16k bytes of EDAC protec ted memory. The authore proposed a whopping 64k bytes of EDAC for MicroSat based on the availability of 64k by 1 CMOS memory chips. Surely. this should be enough memory for such a small satellite!

However. the software coders had other ideas and made it abundantly clear that at least 128k bytes of EDAC memory was needed and even more would be desirable.

The problem. a8 with most things in this roject, is that there just isn't much PC board area to cram ICs onto. and insufficient volume to easily stack more than three PC boards into. The EDAC memory utilizes twelve (12) memory chips to save enough information to recover from a one-bit error in each eight-bit byte.

Fortunately. 256k by 1 CMOS static RAMs became available in very limited quantities by June of 1988. The specifications for the chips were discouraging - they would apparently require a tremendous amount of power while operational. The price was only slightly more intimidating. Measurements. however. indicated power consumption would be manageable in this application. Successful operation of the wire wrap prototype running a program from the EDAC memory in early August of 1988 confirmed the earlier power consumption measurements.

Twe 1 ve Hitachi 6207 256k x 1 ICs are employed as the storage elements in the EDAC memory array. with a combination of 74AC and 74HC logic to provide buffering. syndrome encoding and decoding. and detected error logging. This memory occupies the lowest 1/4 megabyte of the V40 address space.

OTHER MEMORY

OSCAR 11's DCE demonstrated that conventional static CMOS bytewide RAMs would perform satisfactorily in low earth orbit [3]. Such memory requires software protection schemes to detect and errors in stored data. They are unsuited for program storage.

In the MicroSat CPU address space. the middle 1/2 megabyte is filled with such memory. A total of two (2) megabytes are located on the CPU main PC board. organized as four (4) banks of 1/2 megabyte each. Each bank is under control of the CPU and can be independently powered on or off. se1 ec ted for access by the CPU or selected for access by an external experiment. This "dual porting" arrangement allows external devices (such as the Weber State CCD camera) to have high-speed access to memory for storage or retrieval of data as well as communication with the CPU. It also protects the CPU buses from any sort of malfunction by the external device.

Since the CPU controls the spacecraft as well as manages the experiments. CPU health is paramount to mission success. Hence, the dual port arrangement rather than allowing experiment access directly to the CPU's buses.

MASS STORAGE

It was determined that a total of two (2) megabytes of non-EDAC RAM is insufficient to support a PACSAT mission. especially when second-generation missions are flown with higher data rate radio channels. Speeds of 56 kilobits per second are envisioned in the near future for PACSATs, and higher data rates soon thereafter. For this reason. additional memory is required.

On the other hand. many MicroSat mission8 may not require large amounts of memory. in which case the bank-switched two (2) megabytes will suffice. At a review meeting in Boulder. Colorado, in June of

1988, WA4ONG suggested a serial-acce88 mechanism be employed to allow a low-power • high capacity data atorage device to be built on a separate PC board. This idea was adopted.

The mass storage board contains programmable address counters that may be written and read by the CPU. A 24-bit address bus is employed on the unit, allowing up to sixteen (16) megabytes of storage. Present chip8 limit this to 8 megabytes, but by mid-1989 it should be possible to build a 28-bit address bus unit (256 megabytes!) with at least 32 megabyte8 populated on the same size card!

The entire mass storage unit occupies eight (8) bytes of 110 space in the V40 address map. The address counter is auto-incremented following every memory read or write, so block move instructions may be employed to rapidly move chunk8 of data between mass storage and directly addressable CPU memory.

DMA

In order to provide capacity for high-speed data 1inks, DMA techniques are employed in the CPU between the V40 and the 72001 serial I/O devices. Under DMA, the CPU ha8 only to set up the DMA controller and serial I/O chip, then go on to other tasks. After the data is sent or received, the CPU will be interrupted. This greatly reduce8 the processing power needed to manage communications and allow8 for very rapid data link rate8 on multiple channels.

GENERAL LOGIC

Final ly, it should be noted that programmable logic devices are not utilized in the CPU design. Currently available CMOS dev ices are either EPROM or EEPROM based (and thus susceptible to radiation damage) or too slow (Harris fusible link PALs). RAM based device8 (Xilinx) are under investigation for future generations of MicroSats, but appear inappropriate for the present. Bipolar devices consume far too much power.

In 0rder to achieve the requi red speeds with unknown capacitive 1oads, AC 1ogic is generally employed. HC logic was used in the OSCAR-11 DCE and has performed well. AC logic will get its shakedown with the first MicroSats. The designers of the CPU think that the ground bounce spike8 made famous by the designers of al ternat ive pinout logic will not be a problem since such transients have plenty of time to subside at the relatively slow bus speed of the MicroSat CPU.

CONCLUSION

MicroSat provides a low cost spacecraft bus upon which various experiments may be flown. PACSATs will occupy tuo of the first four MicroSats. The MicroSat CPU is flexible and powerful, allowing significant growth in future mission capability without redesign.

It is the authors' hope that, uith the successful launch of the first cluster of MicroSats, Amateur packet radio networking will take another giant step forward.

ACKNOWLEDGMENTS

The authors uish to thank the designers of the PS-186 [4] for many of the ideas employed in the MicroSat CPU. These are Mike Brock, Franklin Antonio and Tom Laf luer. Other ideas came from Tom CLark, Jan King, Bob McGwier, Harold Price and Skip Hansen.

REFERENCES

[1] Den Conners, KD2S, "The PACSAT Project," ARRL Amateur Radio Second Computer Networking Conference, pp. 1-3.

[2] Lyle Johnson, WA7GXD, "The OSCAR-11 Packet Experiment." ARRL Amateur Radio Third Computer Netuorking Conference, pp. 64-67.

[3] Jeff V. Ward, G0/K8KA, "An Analysis of UoSAT-2 DCE Memory Performance," AMSAT-NA Technical Journal, Summer, 1987, pp. 4-12.

[4] Mike Brock, WB6HHV, Franklin Antonio. N6NKF, Tom Lafluer, KA6IQA, "A High Performance Packet Switch," ARRL Amateur Radio Sixth Computer Networkina Conference, pp. 14-37.

RADIX 95 :
Binary to Text Data Conversion for Packet Radio

James G. Jones, WD5IVD
University of North Texas
Department of Computer Science

Gerald A. Knezek, KB5EWV
University of North Texas
Department of Computer Education and Cognitive Systems
Denton, Texas 76203

Abstract

Binary files can prove to be difficult to transfer over the current amateur packet radio network. Radix 95 provides a way to convert data such as compiled programs or graphic images to printable ASCII characters and allow their transfer in Converse Mode. Radix 95 (base 95) is a simple variable length encoding scheme which offers greater efficiency than is available with conventional fixed-length encoding procedures.

Introduction

Transfer of data across the amateur packet radio network usually takes the form of point-to-point **computer** connections or store-and-forward BBS's. In the case of point-to-point data transfer, the TNC (Terminal Node Controller) can be configured in ways to allow the transfer of 8-bit data (converse mode [8BITCONV ON, AWLEN 8, XFLOW OFF] or simply use transparent mode). However, the transfer of 8-bit data through the current store-and-forward BBS (Bulletin Board Sysetm) network is unreliable due to the use of certain 8-bit characters for control. In this case, a message that **assumes** the eighth bit is available for data can cause the transfer to fail.

With the use of available compression programs, 8-bit to 7-bit conversion techniques, and file splitting, it should be possible to transfer 8-bit data across the amateur packet radio network with minimal negative impact upon total network operations.

This paper will focus on Radix 95, a base 95 8-bit to 7-bit file conversion method which offers greater efficiency than is available with conventional fixed-length encoding procedures, and its possible use for sending 8-bit data across the network.

8-bit to 7-bit Conversion

Three common steps are involved in converting 8-bit to 7-bit data and transferring it from one point to another :

1. Translate a sequence of bits into printable ASCII code.
2. Transmit the data
3. Convert the ASCII code back to original form.

The problems facing the transmission of 8-bit data are :

1. How to transmit streams of eight or more binary digits through a network that might have problems handling the data.

2. How to pass certain 7-bit sequences through without having the sequence interpreted as a control character. These characters include flow control (DC1, DC3), padding (NULL, DEL), transfer of control (ESC), or any of the other non-printable 7-bit characters. Any sequence of data bits which could represent a control character should not be sent unless some special provision is made to mask it as a printable ASCII character. [Stone 1984; Brown 1984; Da Cruz 1984-1]

3. How to avoid significant amounts of transmission overhead.

Most data conversion methods achieve 1 and 2, but introduce significant amounts of transmission overhead. Radix 95 produces less overhead than most conversion methods now commonly used.

Overhead

Overhead is the measure of how many extra bits must be utilized to convey meaningful information. Encoding overhead is defined as :

$$O_e = \frac{b_e - b_s}{b_s}$$

O_e = encoding overhead.
b_s = number of bits of meaningful data in the source representation.
b_e = number of bits occupied by the meaningful data after encoding.

Current Conversion Methods

Hex Encoding

Hex encoding views each binary octet as two contiguous 4-bit sequences. Thus each group of four bits is translated into its corresponding hexadecimal character.

The mathematical transformation is from Binary (base 2) to Hexadecimal (base 16). After transmission, the 4-bit sequences are recombined into the original data. Table 1 shows the Hex Encoding scheme.

The encoding overhead for Hex Encoding is 75% [(14-8)/8 = 6/8 bits]. With the addition of the accompanying parity bit, most implementations actually use 16-bits to transmit eight bits of information. The total data encoding plus parity-bit overhead is 100%.

BINHEX for the macintosh is a common program which implements Hex Encoding.

TABLE 1 :
Binary Encoding with Hexadecimal

Bit pattern	Representation
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Character Prefixing

Character Prefixing checks every sequence of eight bits and sends it unaltered if the bit stream corresponds to a printable ASCII character. If the eighth bit is a 1, then a special prefix character (typically &) is sent preceding the character corresponding to the remaining seven bits. If the remaining seven bits represent an ASCII control character, then the character is transformed to one which is printable (by complimenting the seventh bit) and the transformed character is also preceded by a second special prefix character (typically #). The special prefix characters are themselves prefixed for transmission. Table 2 shows the encoding scheme.

TABLE2 :
Binary Encoding with Character Prefixing

Bit Pattern	Numeric Equivalent	Transformation	Transmission
00000000	0	1000000	# @
00000001	1	1000001	# A
.	.		.
oooillll	31	lolilll	# -
00100000	32	0100000	SPACE
00100001	33	0100001	!
00100010	34	0100010	“
00100011	35	1100011	& c
00100100	36	0100100	$
00100101	37	0100101	%
00100110	38	1100110	& f
00100111	39	0100111	'
.		.	.
01111110	li6	1111110	~
01111111	127	0111111	# ?
10000000	128	1000000	& # @
10000001	129	1000001	& # A
.	.	.	.
111~1111	255	0111111	& # ?

The overhead for character prefixing is dependent on the type of file being transmitted. Text files are efficient, since few characters need to be prefixed. For a binary file consisting of randomly distributed I’s and O’s, the overhead is 83.5%. [Abel 1986] Kermit is a popular program using Character Prefixing for binary file transfer. [Da Cruz 1984-2]

RADIX 64

Radix 64 partitions three consecutive bytes (24-bits) into four 6-bit units. Each 6-bit sequence (with 32 is added to avoid control characters) is converted to its corresponding ASCII character. The mathematical transformation is from binary (base 2) to base 64. Table 3 shows the encoding scheme.

The overhead of Radix 64 is 16.7% [(7-6)/6 bits], because a 7-bit ASCII character is used to carry six meaningful bits. When parity is added, the total overhead is 33.3% [(8-6)/6 bits].

Uuencode [UCB 1980], found on most UNIX systems, uses the Radix 64 encoding scheme.

TABLE3 :
Binary Encoding with Radix 64

Bit Pattern	Numeric Equivalent	Representation
000000	0	SPACE
000001	1	!
000010	2	"
.	.	.
111101	61	]
111110	62	^
111111	63	

RADIX 95

Radix 95 uses all printable ASCII characters to carry meaningful code. The 31 printable characters in excess of Radix 64 requirements are used to represent designated 7-bit sequences. In addition, since there are exactly two 7-bit combinations such that the first six bits are the same, there are no first six bits of the extra 31 characters. Each 6-bit segment becomes a 7-bit combination when a 0 or 1 is appended to it. 62 (2x31) characters can actually be used to represent 7-bit segments. Table 5 shows the encoding scheme used for Radix 95. The theoretical overhead for Radix 95 is shown in Table 4.

Table 4 : Theoretical Overhead of Radix 95 [Renka 1987]

Let N6 = Number of 6-bit strings in a random binary data file. (Values of range 31-63)
Let N7 = Number of 7-bit strings in a random binary data file.
(These have lower **6-bit** values in the range O-30 and either 0 or 1 as the 7th bit)

then : $bs = 6(N6) + 7(N7)$
$be = 7(N6 + N7)$
$Oe = \frac{be - bs}{bs}$

therefore : $\frac{7\,[N6 + N7] - [6\,(N6) + 7\,(N7)]}{6\,(N6) + 7\,(N7)}$

$= \frac{N6}{6\,(N6) + 7\,(N7)} = \frac{1}{6 + 7\,(N7/N6)}$

Let p = Probability that a random 6-bit string has a value in the range O-30 **p = 31164**
Let q = l-p **q = 33164**

For a theoretical random binary file :
N7/N6 = p/q = [(31/64) / (33/64)] = **31/33**

By Substituting :
$Oe = \frac{1}{6 + 7\,(N7/N6)} = \frac{1}{6 + 7\,(31/33)} = 7.95\%$

If begun with be = 8 (N6 + N7 **),** to allow for parity, **then a similar computation yields 23.23 % overhead.**

TABLE5 :
Binary Encoding with Radix 95

Bit Pattern	Numeric Eauivalent	Representation
0000000	0	**SPACE**
0000001	1	!
	.	.
.		.
0011110	30.	**>**
011111	31	?
100000	32	@
100001	33	A
	.	.
.		.
111111	63.	-
1000000	64	'
1000001	65	**a**
1000010	66	**b**
	.	
.	.	.
1011101	93	}
1011110	94	~

RADIX 95 Encoding/Decoding [Abel 1986]

Common practice is to view a file as a collection of bytes of characters. By taking the input file as a continuous string of bits, it is possible to break the file into segments of fixed or variable lengths. As long as the segments are kept in proper order during encoding and decoding, the transfer takes place correctly .

First take all 64 **6-bit** combinations as **6-bit** binary numbers, such that all combinations are enumerated by their binary value (000000 = 0, **000001 = 1, etc.).** To make use of the remaining 31 printable character possibilities, the first 31 of the **6-bit** numbers are replaced by **7-bit** numbers **(62 total)** created by including a 0 or 1 as the seventh bit.

To encode, six bits are collected from the input file and are assigned place values in ascending order (1,2,4,8) to make a number. If the number is O-30 inclusive, then the next contiguous bit (0 or 1, with a place value of 2^7) is added to the number. The number is then added to 32 to give the ASCII value of a printable character, which is then written to the output file.

Since it is not possible to determine beforehand the number of bits that will be encoded, there will be between zero and five bits remaining at the end of the file when end-of-file is read. This last (short) bit sequence will be written as a full ASCII character, so a special provision must be made to prevent the decoding program from appending extra O's to the output file.

One extra character is written after the last transmission, representing data from the file. This last character specifies how many bits are to be extracted from the preceding character. Example : if the four bits 0110 (value 6) remain at the end of a file, then the character & (6+32) is written to carry the data, followed by the character $ (4+32) to indicate that only four of the seven bits carried by & are to be extracted upon decoding.

To decode, characters are read from the encoded file one at a time, and converted to a number by subtracting 32 from each. If the number is O-30 or 64-94, then the seven bits of the number from the least to most significant are written. 32 is subtracted from the last character decoded, and the number which results is used to determine how many bits (least to most significant) are to be extracted from the next-to-last character.

Benchmarks

The following are benchmarks for Radix 95 vs Radix 64 [Yu 1987]

BENCHMARKS for Radix 95 (including parity")

TEST FILE	Oriainal Size	Result Size	Overhead
objcode	22924	26830	17.04%
objcode	45848	53658	17.03%
objcode	91696	107314	17.03%
source	41796	50798	21.54%
source	56040	68162	21.63%
random	40000	49340	23.35%
random	80000	98712	23.39%

BENCHMARKS Radix for 64 (including parity")

TEST FILE	Oriainal Size	Result Size	Overhead
objcode	22924	30565	33.33%
objcode	45848	61130	33.33%
objcode	91696	122261	33.33%
source	41796	55727	33.33%
source	56040	74719	33.33%
random	40000	53333	33.33%
random	200000	266666	33.33%

objcode - Object Code generated by C compiler.
source - C source Code.
random - random binary digits.
† - Forced by Disk Storage Method.

Radix 95 encoding for random binary data produced the greatest overhead of the three types of test data included for this scheme. Also as the two tables show, Radix 95 produced less overhead than did Radix 64 in all three test groups.

Discussion

Radix 95 offers greater efficiency than does other commonly available conversion met hods However, encoding overhead is just one of the factors that determines the overall efficiency. Processing time for the conversion must be kept in mind. At some point the cost in file size overhead will be offset by the amount of computation time required. It has been suggested that Radix 95^2 = 9025 paired printable characters or 95^3 = 857,375 triplets **(even more time consuming)** be investigated for further reduction in transmission overhead.

Recommended File Formats

RADIX 95

The following is a proposed standard format for Radix 95 files after encoding :

1. Top line of a Radix 95 file will read :
   (RADIX 95 - [FILENAME : DATE])
   Filename is the first 8 characters of file name
   Date is - 00/00/00 [Month/Date/Year]

2. Each line of Radix produced will be 70 characters long. No other information should appear after the last character on the line.

3. The file ends with :
   (RADIX 95 - END FILENAME).
   Filename is the first 8 characters of file name

4. A Radix 95 conversion program should be designed to search a file's first 10 lines **(one that is to be converted back to 8-bit data)** for the **"(RADIX 95"** before attempting a file conversion. If the program cannot find the correct line within the first 10 lines, then the program should abort.

File Splitting

The following is a proposed standard form for splitting Radix 95 files.

1. The first line of each split file will be :
   (FILENAME.# of #)
   'Filename' is the first 8 characters of the actual file name to be split.
   '.#' is the part of the whole file.
   'of #' is how many different sections.

2. The file ends with :
   (END - FILENAME.# of #)

This will allow the user to specify the first file of a number for the file splitter program to read. Then the file splitter will attempt to use the FILENAME.# convention to reconstruct the file for the user.

User Protocols

In this example, let's suppose KB5EWV wishes to send a data file to WD5IVD.

1. Start with a 65K binary file.
2. Compress the 65K binary file = 39K
   (Assuming 40% compression)
3. Encode the 39K file = 46.8K
   (Assuming a 20% Overhead with Radix 95)
   At this point you have saved 18.2K and you have a completely ASCII file.
4. Determine if the file needs to be split.
   a. **HF SKIP-NET forwarding - break into 5K or smaller segments. That would leave 10 files to be transmitted over a period of days. Remember that the major flow of** SKIPNET **is 300 baud on HF. This accounts for the small size of messages.**
   b. **VHF forwarding - break into 30K or smaller** segments. **Local users will want to consult with their local BBS system operator to determine better message sizes. High-speed networks would be able to handle larger messages than lower speed network connections.**
   c. **Point-to-Point - No need to split a file you intend to send all at one time.**
5. Send File**(s)**.
   Type : Private to KB5EWV @ BBS
   Title of **: RDX - FILENAME.# of # TYPE**
   Type is the data compression program used.

Examples :	**SIT - Mac Stuff-It**
	ARC - IBM ARC
	PIT - Mac Packet-It

6. Recipient receives file**(s)**.
7. Recipient joins file**(s)**.
8. Recipient Decodes **(Radix 95)** file.
9. Recipient removes compression.

Summary

The usage of Radix 95, data **compression,** and file splitting would allow amateurs greater flexibility in sending information involving 8-bit data across the amateur packet radio network. Radix 95 would allow an amateur to load data files onto existing BBS's for local reading or for message forwarding while reducing the amount of traffic over the network compared to if the same files were sent in their original state. Currently there is no agreement on the transfer of 8-bit data. Radix 95 would be an optimal choice in 8-bit to 7-bit conversion for data transfers that require the data be in 7-bit format.

Bibliography

Abel 1986 Abel, J. Alex and Gerald Knezek. Binary to Text File Conversion Using RADIX 95. Department of Computer Science, North Texas State University, 1986.

Brown 1984 Brown, Eric and Art Wilcox. Communications Features Explained. PC World, September 1984, pp. 170-l 77.

Da Cruz 1984-1 Da Cruz, Frank and Bill Catchings. Kermit: A File-Transfer Protocol for Universities. Part 1 : Design Considerations and Specifications. Byte, June 1984, pp. 225-278.

Da Cruz 1984-2 Da Cruz, Frank and Bill Catchings. Kermit : A File-Transfer Protocol for Universities. Part 2: States and Transitions, Heuristic Rules, and Example. Byte, July 1984, pp. 143-l 45, 400-403.

Renka 1987 Renka, Robert. Theoretical Overhead for Radix 95 Encodina. Excerpt from : Binary Encoding Benchmarks Radix 64 vs Radix 95 [Yu1987]. 11 th Annual Computer Science Conference, Federation of North Texas Area Universities. Department of Computer Science, North Texas State University, 1987.

Stone 1984 Stone, M. David. Picking the Proper Protocol. PC Maaazine, vol. 7, no. 4, June 11, 1985, pp. 355360.

UCB 1980 UNIX™ Programmer's Manual 7th Ed, Virtual VAX-l 1 version, November 1980, Computer Science Division, Department of Electrical Engineering and Computer Science. Berkeley, California: University of California, 1980.

Yu 1987 Yu, Carol, Gerald Knezek, and Jeff Carruth. Binary Encodina Benchmarks: Radix 64 vs Radix 95, North Texas State University Department of Computer Science, 1987.

RADIX 95 Source

```
/* RADIX 95 ENCODE   Jeff Carruth & Greg Jones 1988 ●  /
#include <stdio.h>

main()
{
    unsigned long buffer = 0;
    int         temp;
    short        bits = 0;

    while ((temp = getchar()) != EOF) {
        buffer = (buffer cc 8 ) | temp;
        bits t= 8;
        while (bits >= 7) {
            if ((temp = buffer >> (bits - 6)) > 30) {
                putchar(temp + 32);
                bits -= 6;
            }
            else {
                temp |= ((buffer >> (bits - 7)) & 1) << 6;
                putchar(temp + 32);
                bits -= 7;
           }
            buffer &= ~(~0L << bits);
        }
    }
    putchar(buffer t 32);
    putchar(bits + 32);
}
```

```
/* RADIX 95 DECODE   Jeff Carruth & Greg Jones 1988 ●  /
#include <stdio.h>
#define getbyte(a,b,c) (a = b, b = c - 32, c = getchar())

main()
{
    char         current, next;
    int          i, next2;
    unsigned long bit_buf = 0;
    short        in_buf = 0, in-current;

    if ((getbyte(current, next, next2)) == EOF) {
        fprintf(stderr,
            "decode: not enough bytes in input.\n");
        return(-1);
    }
    if ((getbyte(current, next, next2)) == EOF) {
        fprintf(stderr,
            "decode: not enough bytes in input.\n");
        return(-1);
    }

    while (getbyte(current, next, next2) != EOF) {
        in-current = ((current > 30) && (current c 64)) ? 6 : 7;

        bit-buf = (bit-buf << 6) | (current & ~(~0L << 6));
        in-buf += 6;

        if (in-current == 7) {
            bit_buf = (bit-buf << 1) | (current >> 6);
            ++in_buf;
        }
```

Amateur TCP/IP: An Update

Phil R. Karn, KA 9Q

1. Introduction

Amateur radio use of the DARPA Internet protocols ("TCP/IP") has grown from a paper proposal during the "protocol wars" of several years ago to a well-established reality today. Because the TCP/IP software is free and available to radio amateurs and all other non-commercial users, it is hard to say exactly how many are using it. One rough estimate is the number of Internet addresses that have been assigned from the "network 44" block for amateur packet radio: about 1,000 amateurs in several dozen countries. The package has also gained considerable popularity outside of amateur radio, especially in universities.

With the popularity of TCP/IP on amateur radio has come another most welcome development: the appearance of others making substantial contributions to the software effort by creating new features and enhancing existing ones. Several of these contributors have documented their work in other papers in these proceedings, and any other potential contributors are also encouraged to do so. In this paper I will review the TCP/IP developments and experiments of the past year. Although I will mention several contributors by name, the project has grown much too large for this to be an exhaustive list; I hope no one will feel slighted if they are accidentally omitted.

In this paper I will also comment on some of the lessons learned so far, and then discuss possible directions for the future. As expected, much has been learned about the operational aspects of true computer networking on amateur packet radio. We've also learned quite a bit about coordinating the development of a complex software package when volunteers all over the world are involved.

2. New Features) and Enhancements

2.1. AX.25 Support

A complete AX.25 Level 2 implementation has been added to the package. Its primary purpose is to provide hop-by-hop acknowledgement of IP datagrams without having to rely on TCP for end,-to-end retransmission. IP datagrams may now be carried either in connected-mode I frames that are acknowledged at the link layer, or in AX.25 UI frames as before. The default encapsulation mode is set in the configuration file. Individual datagrams can override the default with the type of service (TOS) bits in the IP header.

An optionall "transparent fragmentation" facility breaks up large IP datagrams into a series of AX.25 I frames for transmission over poor links without added TCP/IP overhead. This was implemented as a local extension to the AX.25 protocol.

AX.25 can also be used directly from the keyboard (i.e., without TCP/IP) for communication with ordinary packet stations. Because of the multiplexing provided by the AX.25 Protocol ID byte, "conventiona.!"' AX.25 and TCP/IP/AX.25 operation can take place simultaneously, even between the same pair of stations.

With minor exceptions, the AX.25 code tracks the proposed AX.25 Version 2.1 specification currently under review by the ARRL Digital Committee. The code has been recently rewritten to adhere to the SDL diagrams by K3NA as closely as possible.

2.2. IP-on-NET/ROM

A separate paper in these proceedings by Dan Frank, W9NK, documents an important contribution to the package: the ability to pass IP datagrams through NET/ROM networks.

This is very much in keeping with a fundamental principle of internetworking that accounts for much of TCP/IP's success: the ability to use facilities that were designed by others without internetworking in mind. Dan's paper discusses in detail the specific approach taken and its advantages and disadvantages.

2.3. Packet Driver Support

FTP Software, Inc, has released into the public domain its interface specification for packet-oriented hardware device drivers on the IBM PC. Packet drivers run as Terminate and Stay Resident (TSR) modules. They are loaded independently of the protocol modules that use it (e.g., the net.exe program that implements TCP/IP on the PC), and they can be shared by multiple protocol modules under a multitasker such as DoubleDos or DesqView.

A major practical advantage of the packet driver approach is that the driver software can be developed and maintained completely independently of the protocol code. Since the drivers no longer need reside in net.exe, the latter can be much smaller; users need not waste disk space or memory on drivers for devices they don't have.

As the packet driver specification becomes a de-facto standard, we will be able to use drivers written by others. TRW had already contributed its PC-2000 Ethernet card driver when I added packet driver support to net.exe. Russ Nelson has since written packet drivers for the Interlan NI5120 and 3Com 3C501 Ethernet cards, and Bob Clements, K1BC, has written one for the Western Digital WD8003 Ethernet card.

2.4. Other Drivers

Art Goldman, WA3CVG, and Richard Bisbey, NG6Q, have contributed a driver for the Eagle HDLC interface card. This card, for a time a popular item at swapmeets, features the Zilog 8530 chip. They started with an 8530 driver I wrote for the PACCOMM PC-100 and added many performance enhancements.

2.5. Routing

Al Broscius, N3FCT, has written automatic routing code for the package. The protocol is the widely-used "RIP" (Routing Information Protocol) that has been the informal standard for several years with Berkeley UNIX systems. (Berkeley adopted it from XNS, the Xerox Network System.) RIP belongs to the class of routing algorithms known variously as "Bellman-Ford," "Distance Vector," or "The Old ARPANET Algorithm." NET/ROM's internal routing algorithm is also in this class, although it differs in detail from RIP. Al's contribution is intended primarily for Ethernet LANs where RIP is already common.

2.6. Electronic Mail

The original mail subsystem by Bdale Garbee, N3EUA and myself has been greatly extended and enhanced by Dave Trulli, NN2Z. Dave and Bob Gibson, WA3PXX, have also each implemented automatic SMTPIWORLI mail forwarding systems. This is a particularly useful feature as it allows those running TCP/IP systems to receive "regular" packet mail without having to log into a BBS manually.

2.7. Additional Applications and Features

Several useful applications have also been contributed. Mike Horne, KA7AXD, has written a "finger" server. This is a popular ARPA application that allows one to locate ("put the finger on") a person on a remote system. It is a handy way to make information such as your telephone number or electronic mail address available for the convenience of those wishing to contact you. W9NK has also written an AX.25 "mailbox" that makes the package more useful with conventional packet stations.

I have added several simple features that make it easier to run the code at a remote site. A "remote" server and command allow control stations to reboot the system with new software and/or configuration files, and a "forward" command handles simplex links such as those found in the collision free backbone node described later.

3. Internal Changes

In addition to the new externally visible features that have been described, quite a few changes have been made under the surface. While perhaps not of direct interest to those who simply use the package, these changes are important because they improve performance, reliability and portability, or make it easier for programmers to create new applications.

3.1. Congestion Control

A major contribution to the problem of avoiding congestion in a highly dynamic TCP/IP Internet has been made in the past year by Raj Jain of Digital Equipment Corporation and Van Jacobsen of Lawrence Berkeley Laboratories. Van's suggestions have been widely accepted by TCP implementers, and they are included in my TCP code. I have also contributed to this effort by devising a technique that guarantees the correctness of the round trip time measurements that TCP uses to set its retransmission timer.

3.2. Improved CPU Portability

The ordering of bytes within a machine word, the hardware alignment requirements for various data types, and the exact positioning of elements in a C-language structure all depend on the specific processor and compiler used. These differences can create portability problems. For example, early ports of the package to the Motorola 68000 sometimes generated faults due to the 68000's requirement that short and long integers be at even addresses, something not required by the 8088.

To eliminate these problems, the subroutines that generate and process protocol headers were rewritten to be completely independent of these considerations. Each protocol module contains two conversion routines: one that converts the protocol header from the external (network) format specified by the protocol standard to an internal representation, and another for the inverse function. External representations are always kept as a simple byte string, while the internal representation is usually a C-language structure. With this and other changes, portability to processors other than the Intel 8086 family (used in the IBM PC) has been greatly improved.

The package necessarily contains some 8086 assembler code, despite its inherent nonportability. The assembler code is in three categories:

3.2.1. Interrupt Handlers

Since the various device drivers are interrupt driven, an assembler "stub" fields each interrupt. Each stub establishes a C environment and then calls a C-language function to do the "real" work in handling the interrupt. Although some newer compilers (e.g., Turbo C and Microsoft C) support special "interrupt" functions, they appear to be primarily designed for software interrupt handlers; extra code is still required (either inline or regular assembly code) to make them function as hardware interrupt handlers.

3.2.2. DOS Interfaces

Some assembler code is included because the package uses several DOS system calls that are not supported by the standard vendor-supplied C library. While C passes function arguments on the stack, DOS and BIOS functions accept their parameters in registers; hence the need for special assembler-language inter face functions.

3.2.3. Performance Enhancements

Two functions were: written in assembler, even though they had already been done in C. One routine computes the ones-complement sum of a block of 16-bit integers; this is the ARPA standard checksum algorithm. The other does fast I/O to port-mapped devices such as the 3Com 3C501 Ethernet controller. Writing these routines in assembler improved performance through the use of certain 8086 instructions that are unavailable directly in C. For example, although the checksum algorithm operates on 16-bit integers, the C implementation uses long (32-bit) addition to accumulate the end-around carries. 32 bit arithmetic on the 8086 is relatively expensive because the CPU is a 16-bit unit. However, in assembler the ADC (add with carry) instruction makes it possible to sum each 16-bit word with only two instructions: a LODSW to fetch the word from memory and an ADC to add it plus the carry from the previous addition to the running checksum. Loop "unwinding" further enhances performance.

I usually try to avoid assembler code when possible because of its inherent nonportability. The problem isn't so much in porting to machines completely different from the PC; device drivers tend to be (necessarily) hardware dependent, so they have to be rewritten anyway. The main headache has been nonportability to the various C compilers on the PC. For example, Aztec C provides a macro package that makes it very easy to write C-callable assembler routines. These macros automatically "do the right thing" for each of the 8086's "billyuns and billyuns" of memory models. Unfortunately, a compatible set of macros isn't available with the other compilers, so Russ Nelson contributed quite a bit of time and effort in porting the assembler routines to Turbo-C.

3.3. Improved Operating System Portability

We have tried to improve portability to operating systems and compilers other than MS-DOS and Aztec C. We found this to be much harder than simple CPU portability since file system, native I/O device support and C subroutine library dependencies appeared in widely scattered parts of the package. For example, while the Apple Macintosh, MS-DOS and UNIX all provide hierarchical file systems, the Mac uses the colon (:) to separate path name components while UNIX uses the slash (/) and MS-DOS the backslash (\). Although many of these problems quickly became obvious once a porting effort had begun, it was unexpectedly difficult to identify them in advance. A typical porting effort therefore goes through several iterations with the goal of isolating as much system-dependent code as possible to a small number of files. We have partially succeeded in this, but there are still many #ifdefs (conditional compilation segments) scattered throughout the code. This should improve as the package matures further.

Porting has involved the efforts of several people. Mikel Matthews, N9DVG, did the initial ports to the Apple Macintosh, UNIX System V and the Commodore Amiga. A group in the San Francisco Bay area including Dewayne Hendricks, WA8DZP and Andy Cromarty, N6JLJ has made further improvements and additions to the Macintosh version, and Bob Hoffman, N3CVL is now coordinating changes to the UNIX version. There are undoubtedly others working on porting efforts of which I am unaware.

4. Operational Experience

Most on-air TCP/IP operation to date consists of local "islands" of activity that cover metropolitan areas or regions. The facilities used in each area are diverse; some use IP switches almost exclusively, others use various combinations of analog (FM) repeaters, IP switches, AX.25 digipeaters and NET/ROM nodes. Most activity uses the 1200 baud AFSK format, simply because the equipment is so widely available. The GRAPES group in Atlanta has pioneered the use of the 56 kbps modem by Dale Heatherington, WA4DSY, and TCP/IP has been its primary application.

As expected, the Internet approach has proven highly versatile in adapting to these heterogeneous environments. Once a system has been properly configured, its user merely issues network commands that specify the system with which he wishes to communicate; he need not be continually concerned with the idiosyncrasies of his local network.

4.1. Collision Free Networks

The "collision free backbone" technique described at last year's conference has been implemented in an experimental node in northern New Jersey. Our site receives on two UHF frequencies (431.025 and 440.950 MHz) and transmits on 221.07 MHz; crossband operation allows it to receive and transmit simultaneously. This technique works, although good performance still requires good RF links. It is interesting to hear this site in operation for the first time; instead of the short bursts of useful data interspersed with collisions, idle flag streams and squelch tails one is accustomed to hearing on regular packet channels, the switch's carrier can stay on continuously for minutes at a time during a file transfer, relaying back-to-back data packets and acknowledgements.

4.2. Radio/Wire Internetting

Some experimental linking between different TCP/IP "islands" has been done with GTE Telenet's PC Pursuit service. This inexpensive service provides slow speed (1200 baud) asynchronous communication across Telenet's X.25 network during nights and weekends when excess capacity is available. Since PC Pursuit provides a (logically transparent) asynchronous "bit pipe" between its endpoints, it handles SLIP (Serial Line IP) just fine. A station in each area acts as a gateway, relaying packets between the local packet radio channel and the telephone line. When more than one telephone line and modem is available at a given station, that site can act as a "hub", switching datagrams from one phone line to another as well as between phone line and radio channel. These experiments clearly demonstrate an ability to establish an emergency packet switching network out of ad-hoc combinations of telephone lines and radio channels.

4.3. Politics

Certain TCP/IP groups have unfortunately encountered occasional friction, especially in areas where TCP/IP activity coexists with conventional AX.25 activity. The complaints seem to fall into two categories: channel loading and "garbage characters", but neither problem is unique to TCP/IP. Channel loading is a potential problem wherever interactive traffic competes with file transfer traffic for a low speed channel. One feature of the TCP/IP package that alleviates this problem somewhat with respect to regular AX.25 users is that TCP "backs off" exponentially whenever it loses a packet, i.e., it doubles the time between each successive attempt. This makes TCP far less aggressive on a heavily loaded channel than a regular AX.:25 TNC.

Complaints about "garbage characters" occur when the binary headers of IP or TCP are seen by stations passively monitoring the channel with regular TNCs. Such complaints are not limited to TCP/IP; any high level networking scheme, including NET/ROM, has similar problems, as do binary file transfers. Clearly the correct solution lies with more flexible packet monitoring code in the TNCs, as enough information is available in each AX.25 packet header (specifically in the PID field) to determine what higher level protocol is in use.

5. Work in Progress

This section discusses some of the ideas and goals for the package that are currently being thought about or implemented. These cover a variety of items, both hardware and soft ware.

5.1. Multitasking operating system

Net.exe has long been structured as a "commutator loop" that relies on upcalls and polling. Hardware interrupts simply transfer data between devices and queues that are operated on by routines called from the main loop. (See my earlier papers for further details).

Each application presently provides functions (entry points) that are called asynchronously by the transport protocol modules when external events occur. Applications are not permitted to "busy-wait", and they must explicitly save state in "control blocks" so that each call is interpreted properly. This technique worked surprisingly well for the existing applications (the DARPA FTP, Telnet and SMTP protocols) but it can be clumsy.

With many expressing interest in using the package as a base for more complex applications (e.g., multi-user WORLI-style bulletin boards) I have begun to restructure the package around a multi-tasking kernel. This will provide a simpler programming environment more like that of a conventional multitasking system. Net .exe will remain a single executable file containing all of the applications linked together, and it will still run under a higher level multitasker such as Desqview or DoubleDos.

New "synchronous blocking" primitives more like those in conventional multitasking operating systems will be provided. For example, calls to the network "receive" function will block, if necessary, until data is available. When one task blocks, others (if any) will be run automatically.

The operating system kernel will be non preemptive. That is, once a task gains control it runs until it explicitly gives up the processor by executing a "wait" primitive (e.g., inside the tcp_read subroutine). A CPU-intensive task can also voluntarily relinquish the processor, allowing other tasks to run. without actually waiting for an event. Choosing a non-preemptive kernel greatly simplifies the design of the rest of the system, since "critical sections" are almost eliminated. (Critical sections are those sections of a program where a hardware interrupt followed by a task switch would leave data structures in an inconsistent state, usually causing a system crash. Critical sections can be notoriously hard to find, since interrupts are semi-random external events).

Most multitasking kernels require some assembler language code for saving and restoring a task's registers but the non-preemptive approach combined with a very clever trick suggested by Rob, PE1CHL allows this to be done with the standard C library's setjmp/longjmp mechanism. This technique makes the resulting code much more portable, thereby avoiding the main reason I decided against a multitasking kernel at the beginning of the project.

5.2. Where Is My High Speed Digital?

TCP/IP users feel the limitations of 1200 baud much more acutely than most packeteers, mainly because of the extreme ease with which parallel operations and large file transfers can be commanded. The problem is much like that of running UNIX or some other powerful timesharing system on a PC/XT. Even though a specific program might run just as fast as under a single-user system like MS-DOS, the system quickly runs out of gas when the much more powerful features of the timesharing system are exercised.

Dale Heatherington, WA4DSY, has made an enormous contribution toward solving the raw RF link bandwidth problem with his 56 kilobit MSK modem design. I have built several of these modems, and they certainly work as advertised. On Ethernet, the TCP/IP package transfers a file between a pair of IBM PC/ATs at about 250 kilobits per second. This is by no means blazingly fast by Ethernet standards (the code was written for portability and flexibility, not maximum speed) but it should certainly be able to keep a mere 56 kilobit modem occupied. But Ethernet interfaces are fairly smart devices designed for fast transfers; unfortunately there is as yet no readily available serial interface for the WA4DSY modem with the capacity these modems provide! (Those who saw the original Dayton announcement of the WA4DSY modem may remember its subtitle: "Where, Oh Where Is My High Speed Digital?")

The standard asynchronous serial ports found on personal computers cannot be used directly with Dale's modem as it is synchronous (as is regular 1200 baud packet). The Atlanta group has had moderate success by modifying standard TNC-2s to operate at 56kbps on the radio side, but they are still limited to relatively low data rates (e.g., 19200 bps) between the TNC and the host computer. This prevents full use of the modem's capacity.

It is better to eliminate the TNC entirely and use an interface card on the PC to generate HDLC that can be fed directly to the modem. Several HDLC cards already exist, including the HAPN 8273 card, the PACCOMM PC-100, the DRSI PCPA and the aforementioned Eagle card. However, none of these cards work well at high speeds, because the PC's DMA (direct memory access) facility is insufficient to support these cards without additional hardware. Interrupt driven transfers are entirely appropriate at lower speeds, but at 56 kbps a byte arrives every 142 microseconds; that is not much time on a 4.77 Mhz 8088 to save state, process an interrupt and return. Making matters much worse is that much PC system code disables interrupts for relatively long intervals (consider disk transfers), effectively "starving" a hungry high speed modem.

It is possible to make one of these simple cards operate in half duplex at 56 kbps -- after a fashion. I am presently writing a driver for the DRSI PCPA that will run at 56 kbps, but it requires that interrupts be inhibited during any active packet transfer. Packet transmission and reception are both done with "busy waits"; the transmit routine is invoked whenever a packet is to be sent while the receive routine is invoked by the appearance of a receive carrier. (This has an unfortunate side effect: open the demodulator squelch and the system "hangs" until the squelch closes).

The proper long-term solution to the "Where Is My High Speed Digital" problem lies with hardware: a special "slave" processor board with sufficient intelligence, buffering and autonomy to handle several seconds' worth of packets without immediate assistance from the host processor. Mike Chepponis, K3MC, has been working on the design of such a board, and his paper on this subject appears in these proceedings. His design appears sufficient for speeds considerably faster than 56 kbps as well.

5.3. Routing

Saying that automatic routing is a "fertile research area" in the professional computer science community is like saying that Antarctica has a lot of ice. Much practical experience has been gained from many commercial, military and research computer networks that can now be applied to amateur packet radio, and I would like to offer some thoughts on how the static, manually controlled routing in the TCP/IP package might be replaced.

As mentioned in the section on RIP, many existing networks, including NET/ROM, use a routing algorithm that is variously known as "Bellman-Ford", "Distance Vector" or "The Old ARPANET Algorithm". Each node broadcasts its routing table, the entries of which list each destination in the network and that node's current estimate of the "cost" to that destination. This algorithm is known to have many problems in practice, particularly with susceptibility to routing loops when links fail. Radio creates a few problems of its own, including particularly flakey links and non-reciprocal paths.

A different algorithm, known variously as "Dijkstra's algorithm," "Link State Routing," ''SPF'' or "The New ARPANET Algorithm" may offer an (alternative. In link state routing, each node broadcasts to all other nodes the status of each of its local links. Each node receives the broadcasts of the other nodes and constructs a complete map of the network from which it may make local routing decisions. The broadcast mechanism is in fact implemented with "intelligent flooding", where an incoming packet is relayed by a node to each of its neighbors only if that packet hasn't been previously seen. (This mechanism has other uses, such as the provision of a user broadcast mechanism).

The only special problem radio presents to this algorithm is how to determine whether a link is "up" or "down". Just having an active AX.25 connection isn't enough; it might have been idle for several hours, so there's no way to know if it's still good or not. Continually transmitting link test packets ("pinging") is not a particularly good idea, for fairly obvious reasons. Here an idea from the DARPA packet radio projects should be very helpful. Each node only has to broadcast a periodic report listing the stations it has heard in the last N minutes, with a count of the number of packets seen from each. Each station mentioned in the report can then compare that number against the number it actually transmitted, gauging the reliability of that particular path. If the percentage is high enough, it can consider the link to be "up"; otherwise it is "down". Or intermediate figures representing link "quality" could be used. Clearly there is plenty of interesting and useful work that can be done here.

6. Concluding Thoughts

Bringing TCP/IP to amateur radio has been a very time-consuming task, but its acceptance has been a most gratifying reward. Many challenges still remain, and not all of them are technical. We have: learned and demonstrated many things so far. We have proven that TCP/IP is indeed a practical and versatile foundation for the more advanced amateur packet applications, but we have also learned that many other packeteers do not as yet feel that they need its capabilities. There now seems to be a growing awareness within amateur packet radio that, at present, no one approach satisfies everyone's needs and that there is plenty of room for parallel, complementary approaches. As more powerful hardware becomes popular, though, I do believe that the "computer networking" philosophy as demonstrated by the TCP/IP project will gradually replace the more traditional "terminal networking" systems we have now.

The KA9Q Internet package continues to be available on the same terms: it may be freely copied, used and modified for amateur or other noncommercial use. Commercial use of any part of the package requires permission of the appropriate author or authors.

CELLULAR AREA COVERAGE TRANSPORT NETWORKS

Donald V. Lemke, WB9MJN
1529 Clyde Drive
Naperville, IL 60565
@ WB9MJN

Abstract

With the advent of level 3 software becoming widely available to Ham Packet Radio, a complete rethink of transport radio systems can lead to greater thruput improvements. This paper details a cellular area coverage transport system. Designed to be collision-free, and not significantly effected by modem lock-up times, this radio system would have thruput equivalent to modem data rate.

System Purpose and Considerations

Packet radio needs much higher thruput networks in many parts of the country. In the urban areas, the problem is too many stations on a each multiple access channel. Here, mainly CSMA (Carrier Sense Multiple Access) collisions reduce thruputs on multiple access channels, making them highly unuseable for thru traffic. In rural areas, there may be fewer stations on a given channel, but they are much more likely to be hidden from each other due to terrain or distance, causing the network thruput to follow the ALOHA theoretical predictions.

Many groups in packet have long decided to have a network for transport separate from user access. While this is a step forward, these networks are designed at present to be multiple access as well. This can result in a 3 (worse case CSMA thruput) to 6 (worse case ALOHA thruput) times reduction in thruput at full load. Just when the network needs to be most efficient, it is the least efficient. The functions of the transport network, and the user access node are significantly different, that much thruput can being thrown away by copying the user access station configuration. This is regardless of any data rate increase on the transport network, and simply a matter of statistical effects of multiple access in the radio environment. 1200 baud packet has a thruput of 500 baud with 256 byte packets, and .3 sec TXD. On an ALOHA channel under full load, such as 145.01, this drops to 85 baud. For the full load to become locked in, a thruput demand by several stations of 250 baud or more is all that is required.

Most of the present transport network projects are basically point-to-point. This causes some difficulties with human nature. Many people feel alienated from groups, whose previous history have directed them into building networks that are not close-to-home. Close-to-home is a highly subjective thing. It may mean literally the closeness to ones home QTH, or it may mean the closeness to ones desired direction for the network to go. Many packet users fail to support network construction efforts because of this I believe. A technological solution to this would address area coverage. If a network existed which was almost transparent in transporting data from an area, which is necessarily not immediately close to a point-to-point transport network site, support from that area would then be forth coming. I am continualy amazed at the failure of people who live in river valleys to accept the physics of radio propagation and topology. An area coverage network between the user and the point-to-point network should gain much support from the WIIMBY (Why Isn't It My Back Yard) set, and also help to off-load traffic from the necessarily low data rate (rf margins vs. money problems) point-to-point links.

Much of the packet revolution on ham radio remains unexplored, at least unexplored on-the-air. Multi-user mailboxes and file servers, wide area coverage conferencing, digital voice, graphics, reliable medium range real time communications, are yet to be applied widely. The main reason for this, is that packet thruput is much too meager to support these applications.

Packet resources pulling on volunteer time and effort, have very long lead times till installation. Therefore, it makes sense to try to improve thruput as much as possible per improvment cycle. To this end, this proposal makes no compromises on thruput. The end result is more expensive than other possibilities, but not unachievably so for cooperating ham clubs. The additional expense does pay for a system with more quality than other systems, resulting in a higher thruput/cost ratio in doing the area coverage job.

Single Access

A single access channel is one in which only one transmitter is allowed at any given time. A single access link network would be collision free. Such a network provides the largest thruput, for a given data rate modem.

There are many ways to do single access. Token passing, polling, and frequency multiplexing are the ones that come to mind. In the ham environment, where a variety of protocols exist, the software techniques of token passing, and polling have a political burden associated with them that would probably prevent widespread use of compatible links. Additionally, these techniques do slow thruput slightly over frequency multiplexing.

Frequency multiplexing of link stations would be the ideal way to provide single access links. It can be used with any protocal, and provides the best thruput. Frequency multiplexing is expensive tho, on first glance.

Full Duplex

Full duplex operation, using existing ham modems at data rates 56 kBd or higher, can more than double one-way thruputs. It is expected that higher speed modems will will require the same or more bits to lock onto incoming signals. More sensitive modems require many more lock-up bits. By keying a link transmitter for the duration of a QSO, modulated with sync pulses, the thruput from one node to the next, will be more than double, just because the receiving modem and digital phase lock loop (DPLL) won't need to need to lock up on each frame. For Example the WA4DSY 56 kBd modem takes an estimated 15 msec to lock-up. With the typical 256 byte-packet, this is a thruput of 13 kBd, without considering ack time.

While it may be argued that with widespread busy network use, each packet would rarely be small, this hasn't been observed where the network covers a wide area, and and the entire paths of the simultaneous QSO's are often not coincident. Additionally, network and link layer acks are small frames, and even putting many of them into a packet results in short packets. There is no way to really control packet size, without introducing additional delays for the individual user. Thus, simplex is extremly inefficient for our area coverage transport networks at 56 kBd or above.

With the addition of coherent demodulators, a full duplex link can run at lower power levels than a simplex network. The increased lock-up time of these demodulators would be negligible in a full duplex transport network. Altho at VHF and UHF frequencies it may be cheaper to improve transmitter power, above 1296 MHz this might not be the case. The RF loss of the frequency multiplexing device, commonly called the "duplexer", as compared to the loss of a PIN diode T/R switch, is small. Typically, the duplexer has 3/4 dB more loss than the T/R switch over the VHF/UHF spectrum. This difference in loss is even smaller at microwave frequencies.

The cost of frequency multiplex drops dramatically with increasing frequency. At the same time the bandwidth thru the rf hardware increases and the range per hop thru omni-directional antennas falls.

Examples

In the introduction above I figured that 145.01 under full load would have a thruput of 85 baud. The thruput of a 1200 baud transport network that was both full duplex, and single access would be 1200 baud. This is a 14 times improvement without improving data rate. At 9600 baud the network would be 113 times as fast as 145.01 and 56 kBd would be 659 times as fast. At 224 kBd, the network would be able to handle two channels of 56 kBd PCM, as well as 112 kBd of data thruput at a whopping 2635 times improvement over 145.01! Think about a network that interlinks many of the voice repeaters in an area with dial up digital voice links, as well as hi-speed digital LAN's?! This is so removed from the realm of the present day Ham reality, I hope you don't all think this is a "pipe" dream. The numbers don't lie though, and this could very well be done with a 224 kBd version of the network detailed below.

Cellular Transport Network (Cellnet)

To do area coverage, one needs a mesh of nodes. To do a mesh of nodes with full duplex, and single access channels in frequency multiplex, one can assign each geographical location a frequency of its own. Since there are far fewer frequencies than geographical locations, some scheme for frequency reuse is needed. To reduce costs, each node should have the fewest number of receivers, and thus neighbors. Additionally, its desirable that any two points geographically separated by the same distance will have the same propagation time between them, within the network, as any other two points separated by that same distance.

Area coverage cannot be done with one link per node. It is possible to do area coverage with two links per node by drawing a line with the links, and then rastering back with another line at the boundary of the area. This has three problems tho. First, a station in one raster line would need to send packets to the end of his raster to get to the neighboring raster. Thus, significant propagation delays could be incured by relatively close geographical neighbors. This is less reliable as well, since relatively short range communication requires the participation of many link stations. Third, a network built to cover an existing demand area, might not be easily modified to cover extensions to that area in the future.

A three link per node scheme meets all the requirements. Paths between any two nodes are relatively direct, and area coverage is achievable. If one starts with several three pointed stars with 120 degrees spacing between rays (like the Mercedes-Benz symbol) and makes a pattern with them, there is one arrangement that can repeat itself ad infinitum, without any of the rays crossing. This pattern of hexagons is shown in Figure 2.

Full duplex can be done with this scheme if each node has two independent RF bands and that each node's transmitter is on the opposite frequency band from it's neighbors' transmitters. The two independent RF bands needed per site can be achieved with separate antennas and two ham bands or with a single antenna and duplexer. At UHF and lower microwave bands the cost of **feedline** is more than the duplexer. At VHF, the only available spectrum, that might be used for this is on a single band. For an omnidirectional antenna **cellnet,** a single ham band and duplexer is the best course to acheive the two independent frequency bands. See Figures 2 and 3 for a typical spectrum and geographical layout. The node numbers in the two figures correspond.

The next question is **"How** many discrete frequencies are needed to implement this network?". At a certain range it is safe to reuse a frequency without interference to other net members. I have no common **sense,** or **mathemetical** explanation, but by laying out a large network on a piece of paper, I was able to figure that 5 channels in each of the independent frequency bands was the minimum required. With 5 channels the closest node that transmits on any of the receive channels of a particular node, is 3.4 times the node spacing away from the particular node.

If antenna heights are adjusted to give **grazing or** slightly lower paths, the signal strength of the more distant transmitters should not effect proper reception of node neighbors, since they will be significantly occulted by the bulge of the earth. In troublesome terrains where it may not be possible to mount antennas low enough, **downtilt** and combined beam antennas may be needed. Generally, in most areas, a simple omni antenna will be sufficient. Mounting antennas too low is also a mistake, as the power required to overcome the bulge of the earth reliably, will cause the station to be heard too far away during periods of propagation **enhancements.** This is a common problem on 145.01.

Table 1 details the RF link margins for 220, 440 and 1250 MHz Ham bands and various node spacings. 2 meters and 440 **MHz** are considered the obvious candidates for user access stations which would have to be **colocated** along with the **cellnet** link equipment at the same site. 900 **MHz,** which could have the greatest usage in metropolitan **areas,** is also shared with truck location systems. Due to the interrelation of frequencies within **Cellnet,** it was deemed impractical to set up a network, with the possibilty of the **primary** *use* truck location system coming along and using one of the frequencies. The whole network would then need to be recrystalled, and possibly shifted up or down slightly in frequency, to fit all the channels in the duplexer pass band. Unfortunately, the **neccassary** spectrum in the 220 band to **impliment** a **cellnet** there has **been** stripped arbitrarily from the Amateur allocations in the U.S.. Table **1** was done before this **occured.** A 1250 **Cellnet** could use 440 for user access **,** and a 440'cellnet could use 2 meters. 220 MHz and 900 MHz could be used for rural user **access,** but without a major change in coordination of existing ham and commercial facilities, it would be difficult to use these bands for this in urban areas. Of course, if a 220 repeater owner/operator wishes to change the emphasis of his existing equipment and coordination to packet, the previous sentence does not apply. In the Chicago area, only one or two systems have so changed, and this is too few to support an entire **cellnet.**

Notice in Table 1, that antenna heights are adjusted to give a grazing path when refraction causes the earth's radius to be **4/3** x the actual radius. This and the **assumtion** that the antenna positions at the two sites are at equal heights above the midpoint terrain, simplified the determination of the additional **path loss** due to Fresnel effects using Reference 1, Figure 8. The lengths of feedline, and **thus the feedline** loss was adjusted with these antenna heights. Rooftop installations were assumed to require 50 feet of cable to reach the antenna. The cable loss figures are for **7/8t** inch Heliax **(tm).** The duplexer was assumed to have a 1.5 **dB** one way loss. Antenna noise temperature at 220 **MHz** was assumed to be 10,000 degrees Kelvin, 3,000 degrees Kelvin at 450 and at 1250, 290 degrees Kelvin. These values were determined from the Reference 2, Page 29-2, Figure 1, and are representative of suburban RF environments. The **WA4DSY** modem **was** used for the reference demodulator in these calculations. The **transverter** used in **WA4DSY's** tests has a noise figure of 4 **dB.** Thus, the noise power of his test, assuming a 80 KHz IF noise bandwidth, is -121 **dBM.** Thus for a **1/1000** BER, the RF signal to noise margin needs to be 13 **dB. 1/1000** BER represents a frame of 500 bytes failing to be copied half the time. It is estimated that a 20 **dB** margin above this point will result in the average 500 byte frame being copied 99 % of the time. These details are used in the equations below:

```
Total Path Loss = Free Space Loss +
     (dB)         Fresnel loss + (2 X Cable
                  Loss) + Duplexer Loss


Free Space Loss = 36.6 + 20 log(miles) +
      (as)        20 log(MHz)
```

Fresnel Loss determined from Reference 1.

Cable Loss
@ 220 MHz =.0055(Tower height in feet + 50)
@ 450 MHz =.0080(Tower height in feet + 50)
@ 1250 MHz =.015 (Tower height in feet + 50)
(single station cable loss, multiply by 2
for system cable losses)

Duplexer Loss = 3.0 dB
(includes transmitter to antenna loss at one station, and antenna to receiver loss at the other)

Receiver Noise Power = 198.6 + 10 log(B) +
(dBm) 10 log(Te)

B = Bandwidth in hz = 80,000

Te = Receiver Noise Temperature (degrees K)
= Tant + (LF-1)Tamb

Tant = Antenna Noise Temperature (degrees K)

= 10,00	@ 220 MHz
= 3,000	@ 450 MHZ
= 290	@ 1250 MHz

L = Rx duplexer loss + Rx station cable loss
@ 220 MHz
= 1.5 + .0055 (Tower height in feet + 50)
@ 450 MHZ
= 1.5 + .0080 (Tower height in feet + 50)
@ 1250 MHz
= 1.5 + .0150 (Tower height in feet + 50)

F = Receiver Noise Factor
= Antilog (Noise figure/lo)
= 2.5 (for 4dB NF)

RF power for 20 dB margin = Receiver Noise Power + Modem 1/1000 BER S/N + 20 dB + Total Path Loss - TX Antenna gain - RX Antenna gain

Modem 1/1000 BER S/N = 13 dB
(for WA4DSY modem)

Tx Antenna gain = RX Antenna Gain

= 8.1 dBi	@ 220 MHZ
= 11.1 dBi	@ 450 MHz
= 11.1 dBi	@ 1250 MHz

Based on Table 1 and practical considerations, three versions of the Cellnet look promising. Rural versions of Cellnet using 45 miles spacing and 220 or 450 MHz at 56 kBd. An urban version at 1250 Hz using 56 to 224 kBd at 15 mile spacing provides enough bandwidth for expected thruput increase and the possibility of using 220 or 440 MHz 9600 baud or 19.2 kBd user access. The rural version would need an antenna with a height of 250 ft above average terrain. Sites with this height are much more common than sites that could support longer distances. At 220 MHz 70 watts with the WA4DSY modem and a 6dBd antenna is needed. At 440 MHz, 40 watts with the same modem, and a 9 dBd antenna. The urban version at 56 kBd would need essentially tree clearing antenna heights, and 12 watts. The 224 kBd version, assuming the modem is coherent detection, would need 24 watts and if not coherent, 48 watts. All these powers are for 20 dB margin, Since we have the ability to retry bad data in packet, and the cellnet concept allows easy rerouting if there is a link outage, this should be sufficient margin.

The 220 MHz Cellnet is now impractical with the short-sighted reallocation of the lower 2 MHz of that band.

The MACC (Mid-America Coordination Council) is recomending a duplex link band from 440 to 442 and from 445 to 447. Rural Cellnet would need a 1/2 MHz in each of these bands. Unfortunately, ATV is commonly used in this spectrum. Hopefully, ATV operations can be moved to the two 6 MHz channels between 420 and 432 MHz. This means that the many present ATV repeaters would not be able to operate full duplex anymore. The impact of the FCC decision is now reaping its practical consequences. It would be quite a technical challenge to put a full duplex ATV repeater on the air using the two neighboring ATV channels, but it can be done. Seperate Tx and RX sites a:nd a link between the two would be necessary. Additionally, 438 to 440 could then be used for the control links from the 220 to 222 spectrum displaced by the recent FCC action.

Two 2 MHz bands are (available in the digital portions of the 1250 MHz ham band. 1298 to 1300 MHz in the h.i-side band, and 1249 to 1251 provide the best spacing from other band users, and conform to the 1988-1989 ARRL repeater directory listing of suggested usage. These bands would be broken into the 5 Cellnet channels, each channel being 400 KHz. There is still plenty of spectrum in the ARRL suggested digital allocation for point-to-point digital links. Appendix 1 details a realizable local oscilator scheme to support the 1250 MHz Cellnet bands. An aside; curves for the duplexer mentioned in Appendix 2 show it has a 2 MHz bandpass, which is enough for five 224 kBd channels.

Figures 4 and 4a show two general schemes to do Cellnet. While these are achievable now, RF and computer development could reduce the cost of these implementations.

Appendix 2 details a 56 kBd, 1250 MHz, 15 mile node spacing RF hardware complement, and its cost. $1400 dollars to talk 15 miles away may seem like a lot. Much of this stuff can be scrounged tho. Antennas and duplexers are good projects to be home made, but extensive testing would be required on working prototypes to ensure reliability. Much of the cost is in feedline, connectors and the duplexer.

While the figure shows the computer system that is available today off-the-shelf, most computer people I show this to, frown, go back in their closets, pull out a dusty **Multibus (tm)** card, and **say"Let** me at **it"!** Franklin Antonio and friends' PS-186 switch would be an ideal **Cellnet** controller. The Chicago Area Packet Association **(CAPRA)** has been working on a 68000 based Multi-bus card general purpose packet switch project for a long time. This project could be used as a **Cellnet** controller. A **Cellnet** controller from whatever source, should drop to $500 or less if/when it is fully developed. **I** remember buying a $400 dollar TNC 1 kit, not so long ago, so **I** have good hopes that this kind of cost reduction will occur.

Cellnet and Refraction

While the above calculations predict the average path loss to the neighbor nodes, the path losses can vary widely. During extremes of propagation the signal strengths of the nodes that reuse the frequency even tho they are far over the geographical horizon, **may** not be insignificant. There are various remedies and techniques that can insure that enhnaced propagation has minimal effect on network reliability.

Coherent demodulation has better capture effect than noncoherent demodulation. Steve Goode, **K9NG,** said that one version of an MSK modem he was working on had a -7 **dB** interference tolerance, in coherent form, but only a -14 **dB** in non-coherent form. While this modem was not identical to the **WA4DSY** modem, it shows that the coherent demodulation has a much improved capture effect. This is an important feature in a **cellnet** system in overcoming tropospheric refraction variations, commonly called local enhancement openings. Although not the worse **case,** we can estimate that during such **openings,** the earth is propagationally flat for the distances involved. In free space, the signal 'strength difference between the desired signal, and the closest reuse of this frequency at a distance of 3.4 x the node spacing is 10.6 **dB.** Using Reference 1 Figure 2, we see that for a 15 mile node spacing and 28 foot high antenna, the difference is about 20 **dB** for flat earth. These figures must be viewed as only rough estimates. Blockage effects could improve or degrade the ratio significantly. If **siteing** is done so that no blockage occurs between neighboring nodes, then the ratio will be improved.

Super-refractive conditions also occur. During these periods, refracted and direct signals from the desired station may cause destructive interference, reducing its signal strength, while increasing the signal strength of the interfering signal. Luckily, for **Cellnet,** these periods of time are a small fraction on a yearly average. Switching to packet length transmissions, and **CSMA** may provide better thruput during super-refraction episodes. Dynamic routing should prevent any node from losing contact with the network. With time and effort, all these conditions should be able to be handled automatically by the site controller computer. Observation of what each demodulator is decoding and the channel signal strength, qualified by DCD should provide sufficient information. When strong data signals occur, and no packets are decoded on a channel, a site controller would ask the neighbor on that particular channel, and the interfering station to switch to **CSMA.** The interfering station could either be observed after the neighbor switched to **CSMA,** or deduced from programmed network knowledge. Use of coherent demodulation should greatly reduce the periods of time these measures would be necessary. Field tests may show that with coherent demodulators, the above techniques are unnecessary.

User Access.

As the **Cellnet** transport network has no direct RF access by users, a data radio system on a separate, and independent channel is required for this. A variety of user access schemes are available. The site controller then combines the users, traffic into a single stream for transport thru the **Cellnet** radio system.

For the rural **Cellnet** scheme, I believe a 2 meter 1200 baud, AFSK, half-duplex, regenerative digipeater would be best. This scheme would use an RF arrangement similar to the average voice repeater. Incoming demodulated data would be squared-up before applying it to the transmitter's modem. The **Cellnet** controller would be **or'd** into both Tx and Rx data streams, and would have logic to inhibit transmissions when the Rx channel was busy. Such a system would eliminate hidden station affects, a big problem with **scatered** users, all with beam antennas pointed at the site. It would probably provide the first LAN style operation many rural **packeteers** will have experienced. The shift from ALOHA to **CSMA** thruputs, and the absence of data resend time would quadruple rural LAN thruputs. All this without any modification of user equipment.

For urban **Cellnet,** I like 440 MHz, 9600 baud **K9NG** simplex user access. Since the cells are relatively small, most users will DCD each other, and CSMA is in effect. 9600 baud **K9NG** modulation should be able to operate in the 25 KHz channelization on 440. Radios with 15 to 20 KHz IF filters are needed. User radios for this would need to be older crystalled radios. Most new PLL radios do not have quick enough turnaround. Surplus commercial radios such as Motorola MAXAR, FLEXAR, and **MICOR** might be used if the IF bandwidths are wide enough. When 224 **kBd** transport nets are in place, 900 MHz, **56kBd, WA4DSY** simplex may be more appropriate. Following the same logic as with rural **cellnet,** only 5 channels for user access would be needed.

GLB radios should be easy to modify for **K9NG** modulation. AEA radios could be used for this too. **It's** really a shame, that these companies decided on conflicting modulation standards, when the **K9NG** standard has been in place for such a long time. Now, there are very few hi-speed packet stations.

A newcomer to high speed packet feels he needs to be a modulation expert, just to buy a radio. Its just a fractionalization, packet can do without and everybody has lost. TAPR could have helped to prevent this too. Think where we would be-now if the TNC2 had 2 versions, the original, and a version with a K9NG modem on it! Yep, that s right, the K9NG modem is older than the TNC2! It really is unfortunate that many people did much to create a perception of imperfection surrounding the K9NG modem, which is completely unfounded. This was confused by many others to include the K9NG modulation standard as well. The K9NG modem works well when matched to the radio. In this respect, the K9NG modem is no different than the original TNC2 AFSK modem. The filtering in the radio and modem needs to be designed correctly, with proper consideration of the receiver IF bandwidth. Now we have two reinvented wheels, and a hard choice between the three modems. An improved modem redesign is now being done by our British compatriots, instead of American companies. The use of a switched capacitor filter, and resistor header has been apparent to us as the best way to get around the various IF bandwidths of off-the-shelf radios and use of the modem at other baud rates, for sometime. The British project should result in good modem performance improvements. End of Soapbox speech. I have to admit, not many thought that the TNC2 with 5 MHz clock would work at 9600 baud, but several are in operation around Chicago, using K9NG modems, with either Howie version 1.1.5 or Net/ROM.

Digital Audio Applications

Many repeater systems have extensive auxiliary links to support extended operating ranges. Frequencies used for this burden our allocations. Now, I'm not talking about the repeater that uses one auxiliary link to link its receiver to its transmitter site. I'm talking about the repeater that has 2 or more remote receivers and a 2 meter transmitter high enough and powerful enough to let HTs hear it in all of the remote receiver zones. 'With several repeaters connected by Cellnet in dial-up fashion, a large coverage area can be had for each repeater% users, without each repeater needing its own set of auxiliary link channels and the powerful transmitter. Although wide area coverage would not always be available, as somebody might be using the target repeater, the ability to talk thru ranges much farther distant than an independent link system could provide, makes up for this problem. Its conceivable that during low usage hours, with an HT, one could easily talk anywhere in a metropolitan area, with a rubber duck and without any propagation enhancement. This capability here in the Chicago area now, requires a 40 foot high antenna, and 40 watts. Generally tho, only one, or two Cellnet hops links could be guaranteed. Don't get me wrong now, even tho Cellnet would be ideal for a repeater without any remote receivers and auxiliary links, repeaters with links would still benefit by gaining the capability to talk to and thru neighboring repeaters.

Error corrected .PCM voice can be sent with differential coding in 24 kBd of audio bits. Some overhead for error detection would be needed, Uncorrected PCM needs 56 kBd audio bits, and no overhead. Thus, 112 kBd could probably support 2 or 3 audio channels. Uncorrected PCM has the advantage, that a level 4 protocol would: not be needed to eliminate hard-to-u.nderstand gaps in the audio. With developement and a level 4 protocol for differential PCM. To eliminate the gaps, this might gain an additional audio channel and make all channels distortion free.

Conclusions

The Cellnet concept is the best way to proceed from this point in time onward for the development, and implementation of packet transport networks. It is .a no compromise solution to our biggest problem, and is also the system of maximum quality In metropolitan areas, point-to-point bypass links will be still be needed, but rural cellnets as described would have equivalent performance as rural terrestrial point-to-point up to 56 kBd.

There is significant development to be done to improve Cellnets' wide spread implimentation. Luckily, many of the packet projects of recent months are directly applicable to Cellnet systems, with small software changes. An RF hardware prototype is not assembled as of this writing in August 1988. IMD may change the designs in figures 4 and 4a. Specific hardware that can probably be made into an assembled Cellnet site is identified, and we are about to begin purchases.

Packet clubs on average are meager affairs. With a few exceptions, packet clubs are not going to be able to fund a complete Cellnet over their memberships area. On the other hand there are many rich repeater and general interest ham clubs with lots of hamfest receipts sitting in their banks. Much of this goes to fund the next year's hamfest, but a lot sits and accumulates. Much worthy support for Cellnet can be had if some of the Cellnet thruput is applied as audio link channels.

References:

1) "Radio Propagation Fundamentals," Kenneth Bullington; Bell System Technical Journal, Volume XXXVI, #3, May.1957

2) Reference Data for Radio Engineers, 6th Edition, Copywrite 1977, Howard W. Sands & Co., Inc.

3) "Modifying the Hamtronics FM-5 for 9600 **bps** Packet Operation," Steve Goode, **K9NG;** Fourth ARRL Amateur Radio Computer Networking Conferance, March 1985, American Radio Relay League, Newington, CT, USA

4) "A 56 Kilobaud RF Modem," Dale A. Heatherington, **WA4DSY;** ARRL Amateur Radio 6th Computer Networking Conferance, Copywrite 1987, The American Radio Relay League, Inc., Newington, CT, USA

Appendix 1:

Local Oscillator Scheme for 1250 MHz
Cellular Packet Radio Network (**Cellnet)**

Here is a mixing scheme and local oscillator frequencies to do 1250 MHz **Cellnet** with 2 MHz bandwidth duplexer bands centered on 1250 MHz and 1299 MHz portions of the 23 cm amateur spectrum. This idea can be implemented with a L.M.W. model # UNLV02 circuit, with the addition of a X 2 tap and buffer amplifier. In the **Cellnet** concept, each site transmits on one duplex band, and receives on the other. Thus, neighboring stations to a particular site receive and transmit on the opposite duplex bands. Consequently, the mixing schemes below are bidirectional, with the indivdual sites using the appropriate transmit and receive bands, to suit the local network enviorment. This scheme is designed to use the maximum duplexer bandwidth available of typical **BP/BR (BandPass/BandReject)** cavity duplexer sets. A **BP/BR** cavity set helps provide protective filtering for the wide bandwidth converter circuits. The 2 MHz could be broken up into a variety of channelizations. Other investigations have shown that for a **Cellnet** system, 5 channels are required. Thus, the maximumn thruput scheme would be 5 by 400 Khz channels. Such bandwidth could support 200 **KBaud** data rate.

```
Low     25.0 ---  X - 155.286  - X - 1249
Channel to 27.0 !     157.286    !   1251
                !     BPF        1
91.143 -X2- 182.286 -X3-X2- 1093.714
L.O.    Chain   !                !
                1                1
High    23.0 ---  X - 204.286  - X - 1298
Channel to 25.0       206.286        1300
                      BPF
```

The **WA4DSY** modem/exciter should work with this scheme with a change or two in some coil values. This scheme could also work with **K9NG** modems and commercial versions of Hamtronics receiver and exciter circuit boards at the 156 and 205 frequencies, Both signal paths need to be highly shielded from each other and any RF equipment operating in the site's vicinity on the 1st IF frequencies. This would not be a **catastrophy** unless a local transmitter happened to coincide with one of the 3 receive channels within the particular site's **receive** IF. The channels are 80 KHz wide for the **WA4DSY** modem, and 20 KHz for the **K9NG** modem. With a metropolitan coverage net, as CAPRA is planning to use this scheme for, the site spacing is roughly 15 miles, so antenna heights only need be 30 feet above average terrain or tree clearing height, whichever is highest. Thus new sites should be easy to procure if one does not work out. The **IFs** of this scheme are outside the ham bands, thus this scheme allows full use of these for whatever other purposes at a particular site.

Appendix 2:

RF Equipment Specifications for 1250 MHz
Cellular Packet Radio Network **(Cellnet)**
Don Lemke - June, 1988

Fully Neighbored Site

Antenna:

* Larson # **FB3-1290** - Omni-directional, **9dBd** gain, Fiberglass radome, $120.00

***Cablewave # FLC78 - **7/8ths** Inch Dia. **Heliax (tm)** coaxial cable, 50 feet needed per site, $196.00 **($3.92/foot)**

***Cablewave # **NM78CC** male - N Male Coaxial Cable connector; $108.00 (2 @ $54 ea.)

Duplexer:

** **Tx/Rx** Systems # 28-97-010 - **12** MHz min. spacing,.9 **dB** loss, 200 KHz min. notch?, 2 MHz min. **bandpass @** 1.9 **dB** total loss $371.25

Radio system:

* L.W.M. Electronics LTD. # UNLV02 - Local **oscilator** system, **incl. xtal,** Freq = 1093.716 MHz $95.00

* L.W.M. Electronics LTD. # **1296PRM2** - Receive preamp and Mixer. $93.38

* L.W.M. Electronics LTD. # **1296TMA3** - Transmit Mixer and Amplifier, 1 watt output **power,** $127.46

** Pauldon Associates
#M57762 - 23 cm band power amplifier, 20 w. max, 18 w. for 1w. drive, MINUS heatsink, 7809 auxiliary bias regulator and connectors, $106.00

* Homebrew - T/R Second IF subsystem, LO driver, Rx splitter, parts for power amp, Estimated cost - $60

* Crystals - 3 receive, 1 transmit, $60.00

* Bud # CU 347 - Shielding Enclosure, Estimate 3 per site, $40.05 (3 @ $13.35)

Notes:

* Purchase these items first, this equipment required to start rf and computer development.
** Purchase these items after initial rf and computer development is done, Needed to develop full duplex mods to rf equipment, and test computer with duplex thruputs.
*** Final equipment purchases prior to installing equipment at a working site.

Vendors:

Larson:

Larson Electronics Inc.
11611 N.E. 50th Avenue
P.O. Box 1799
Vancouver,WA 98668

Cablewave:

Nemal Electronics Inc.
12240 NE 14th Avenue
N. Miami, FL 33161

TX/RX Systems:

TX/RX Systems
8625 Industrial Parkway
Angola, NY 14006

L.M.W. Electronics U.S.A distributor:

Down East Micorwave
Box 2310
RR 1
Troy, Maine 04987

Pauldon Associates:

Pauldon Associates, W2WHK
210 Utica St.
Tonawanda, NY 14150

Bud:

Many vendors thruout the U.S.A.

Estimated RF equipment cost: $1400.00

TABLE 1

Cellnet RF Margins

220 Mhz - 6 dBd antenna

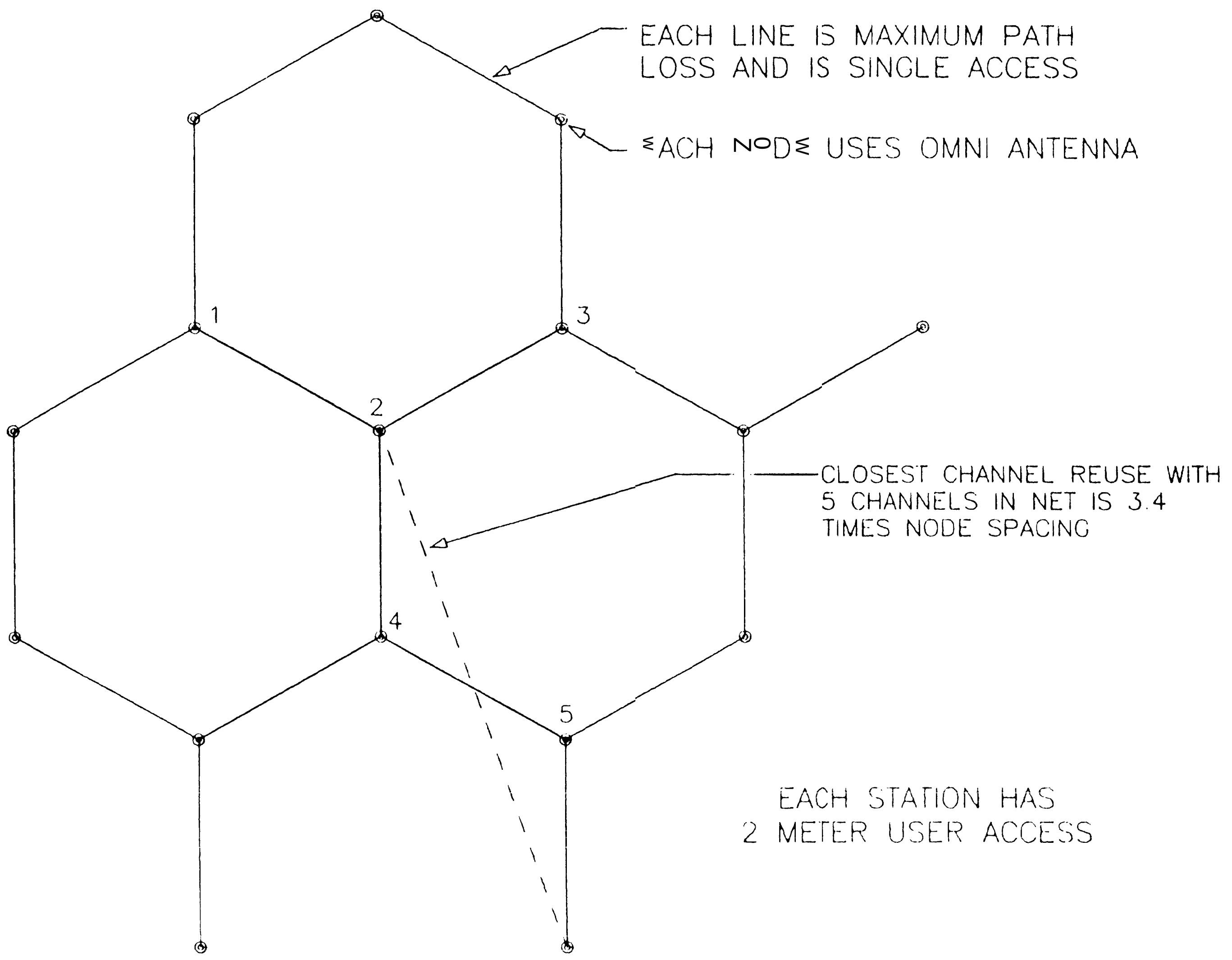
Distance (miles)	F.S.L. (dB)	Antenna Height (feet)	Cable loss (dB)	Fresnel loss (dB)	Total loss (dB)	Te (K)	Nrx (dBm)	RF power (dBm)
25	111.5	78	.70	22.8	138.7	10914	-109.2	46.3
30	113.0	113	90	21.3	139.1	10969	-109.2	46.7
35	114.4	153	1.1	19.8	139.4	11030	-109.2	47.0
40	115.5	200	1.4	18.8	140.1	11122	-109.2	47.7
45	116.6	253	1.7	17.9	140.9	11224	-109.1	48.6
50	117.5	312	2.0	17.1	141.6	11334	-109.0	49.4
60	119.1	450	2.8	[illegible]	143.6	11662	-108.9	51.5

450 Mhz - 9 dBd antenna

Distance (miles)	F.S.L. (dB)	Antenna Height (feet)	Cable loss (dB)	Fresnel loss (dB)	Total loss (dB)	Te (K)	Nrx (dBm)	RF power (dBm)
25	117.7	78	1.0	20.8	143.5	4006	-113.5	40.8
30	119.2	113	1.3	19.2	144.0	4090	-113.5	41.3
35	120.6	153	1.6	18.1	144.9	4189	-113.4	42.3
40	121.7	200	2.0	17.1	145.8	4334	-113.2	43.4
45	122.8	253	2.4	16.3	146.9	4499	-113.0	45.7
50	123.7	312	2.9	15.5	148.0	4708	-113.4	46.0
60	125.3	450	4.0	14.2	150.5	5282	-112.3	49.0

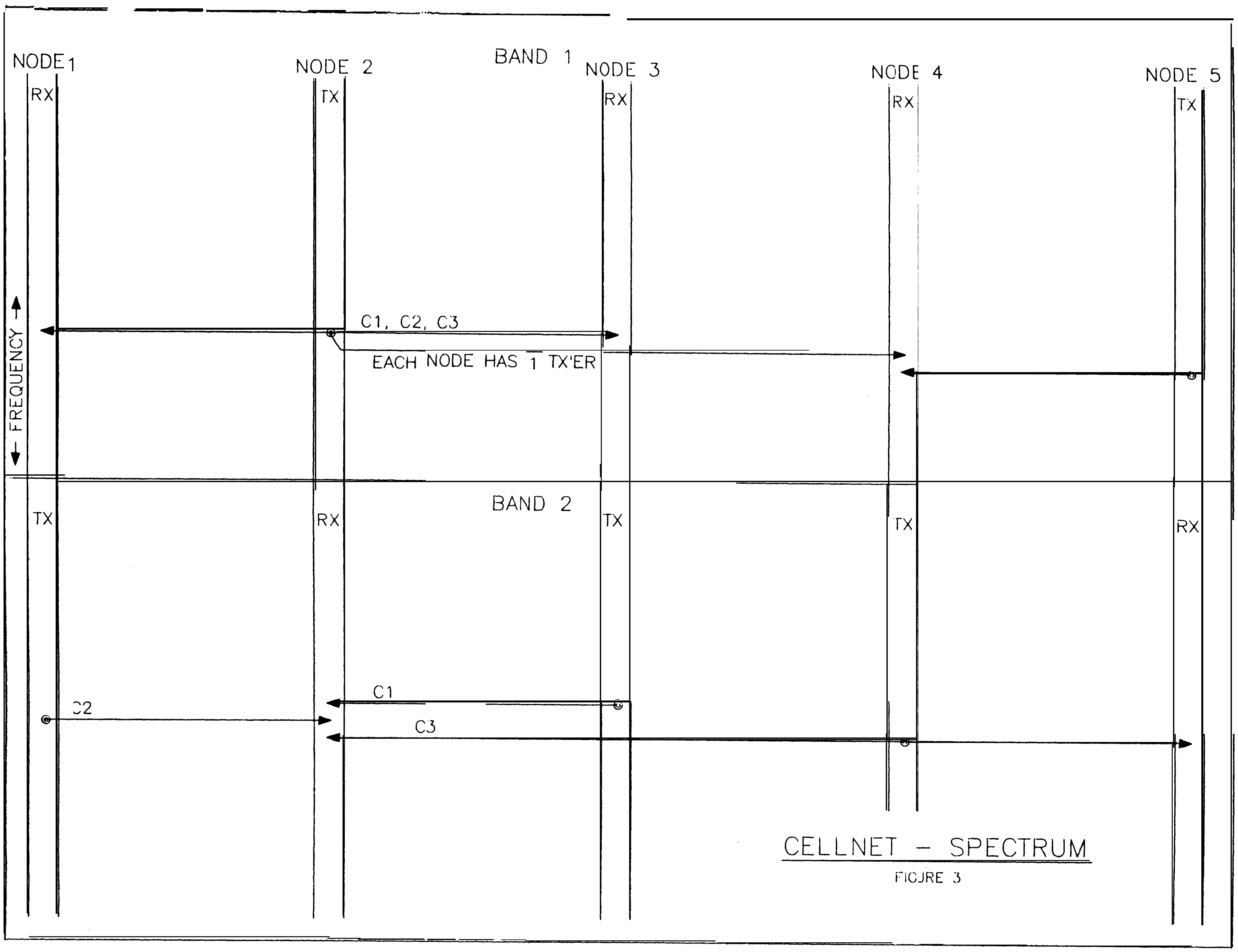
1250 Mhz - 9dBd antenna

Distance (miles)	F.S.L. (dB)	Antenna Height (feet)	Cable loss (dB)	Fresnel loss (dB)	Total loss (dB)	Te (K)	Nrx (dBm)	RF power (dBm)
20	124.9	50	1.5	19.7	150.6	1447	-118.0	43.4
r.t.			.75		149.1	1217	-118.7	41.2
15	122.4	28	1.1	22.0	149.6	1319	-118.4	42.0
r.t.			.75		148.9	1217	-118.7	41.0


EACH LINE IS MAXIMUM PATH LOSS AND IS SINGLE ACCESS
EACH NODE USES OMNI ANTENNA
1
2
3
4
5
CLOSEST CHANNEL REUSE WITH 5 CHANNELS IN NET IS 3.4 TIMES NODE SPACING
EACH STATION HAS 2 METER USER ACCESS

CELLNET – AREA COVERAGE

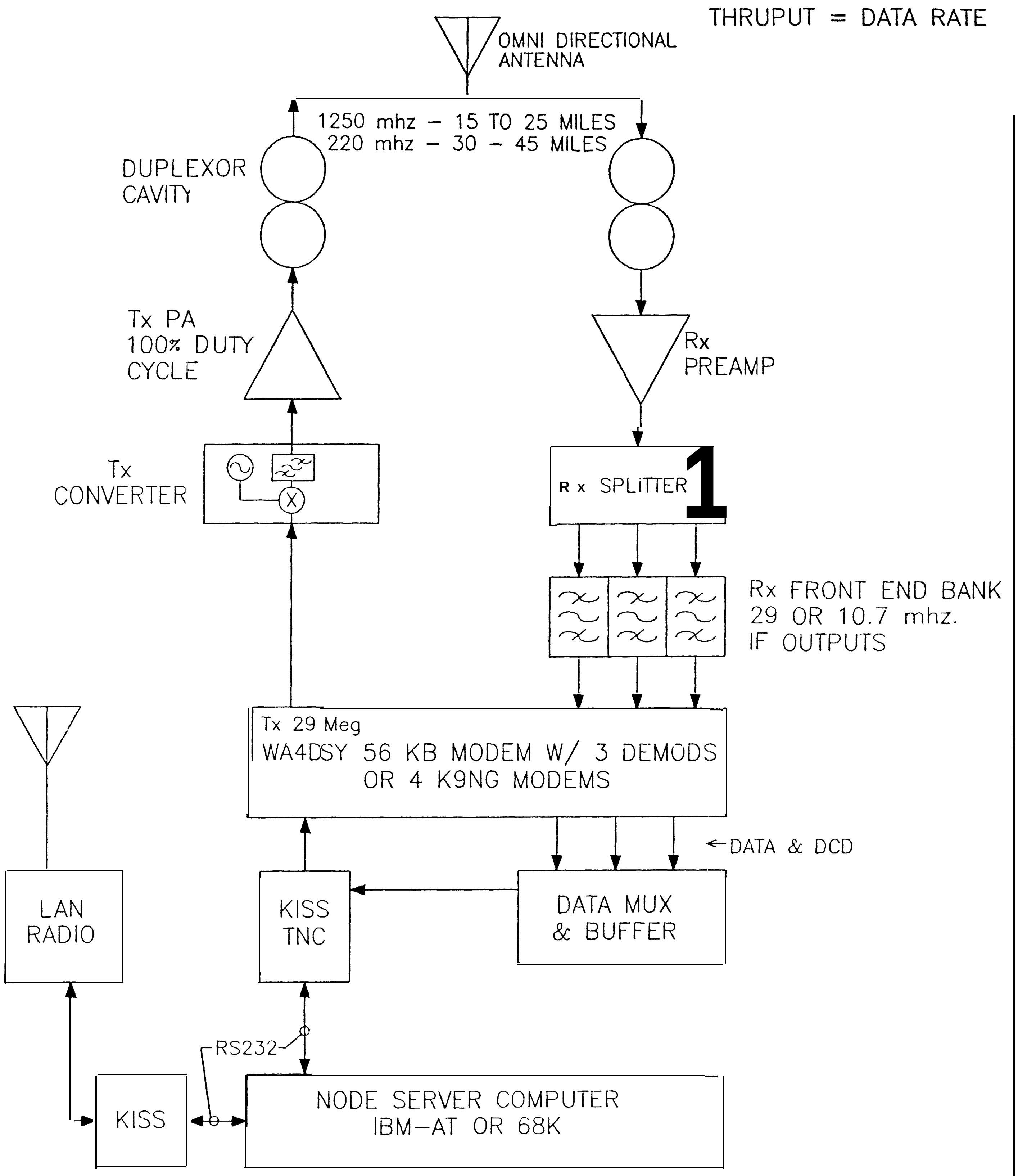
FIGURE 2



CELLNET - SPECTRUM

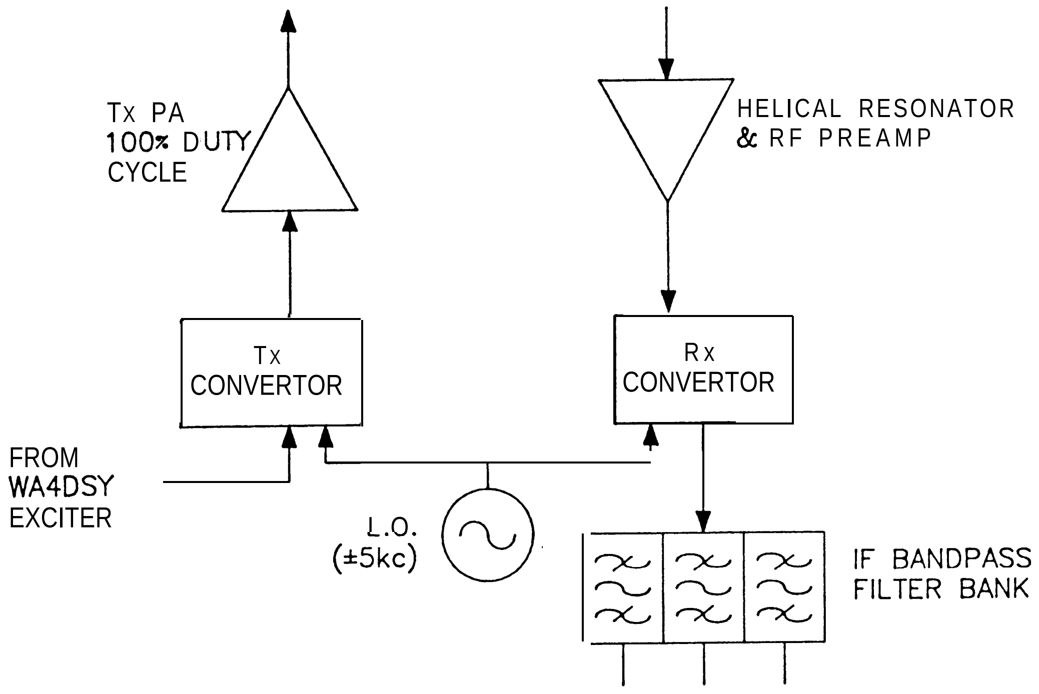
FIGURE 3

CAD REF. CELLNET6



CELLNET - FULL DUPLEX EQUIPMENT

FIGURE 4



CELLNET – **FULL DUPLEX**

ALTERNATIVE RF EQUIPMENT

FIGURE 4a

9600 Baud Packet Radio Modem Design

James Miller BSc, G3RUH
3 Benny's Way
Coton
Cambridge
CB3 7PS
England

ABSTRACT

The theoretical minimum audio bandwidth required to send 9600 baud binary data is 4800 Hz. Since a typical NBFM radio ha6 an unfiltered response from zero some 8 kHz, transmission of 9600 baud binary data i6 perfectly possible though it. This paper describe6 a successful implementation.

INTRODUCTION

The standard packet VHF/UHF radio data rate is 1200 baud because all TNCs provide an internal modem for this speed, and the two-tone AFSK audio spectrum suit6 unmodified voiceband radios comfortably. However, all TNCs can generate much higher data rates, and most FM radios have an unrealised audio bandwidth of some 7-8 kHz or more. So in many cases 9600 baud radio transmission is entirely practical with them.

This design is a high performance full duplex modem designed for packet use with most voiceband NBFM radios, assuming only minor modifications.

A key feature of this modem is it6 digital generation of the transmit audio waveform. Precise shaping compensates exactly for the amplitude and phase response of the receiver. This results in a "matched filter" system, which means that the received audio offered to the data detector has the optimum characteristic ("eye") for minimum errors. It also allows very tight control of the transmit audio bandwidth.

MODEM FEATURES

Here is a summary of the modem features.

* MODULATION: FM. Audio applied direct to TX varactor. +/- 3 kHz deviation gives RF spectrum 20 kHz wide (-60db). Fits standard channel easily.

* 'IX MODULATOR: 8 bit long digital F.I.R. transversal filter in Eprom for transmit waveform generation (12 bit optional). Gives "brick-wall" audio spectrum. Typically -6 db at 4800 Hz, -60 db at 7500 Hz. Allows compensation for receiver (the channel) to achieve perfect RX "eye" . Up to 32 TX waveforms, jumper selectable. Output adjustable 0-8v pk-pk.

* SCRAMBLER (Randomiser): 17 bit maximal length LFSR scrambler, as per K9NG system, and UoSAT-C. Jumper selectable Data or BERT (bit error rate test) mode.

* RX DEMODULATOR: Audio from receiver discriminator, 50mv-10vpk-pk. 3rd order Butterworth filter, 6 kHz. Data Detect circuit for use on simplex (CSMA) links. Independent unscrambler.

* CLOCK RECOVERY: New digital PLL clock recovery circuit with 1/256th bit resolution. Average lock-in time 50 bit6 (depends on SNR).

* CONNECTS to Ax.25 TNC "Modem disconnect" jack. Suitable for TNC-2 (and any other provided the internal modem can be bypassed). Standard TNC digital connections needed: TXData, TXClock (16x bit rate), RXData, Data Detect ("DCD"), GND, (RXClock available). TTL levels. RADIO: TXAudio, RXAudio, GND. All connection6 via 0.1" pitch pads for SIL connectors or direct soldering Unwired DIN 41612 96-way connector (use optional).

* POWER CONSUMPTION: 10 to 15v DC at 40ma (CMOS Roms), 170ma (NMOS Roms). Total 19 ICs (13 CMOS, 2 DACs, 2 op-amps, 2 Eproms). 5 volt regulator and heatsink.

* OTHER FEATURES: The only set-up is TXAudio level. Channel calibration facility. Audio loopback. No hard-to-get parts.

* PCB: 160x100mm (single Eurocard format). Top professional quality, double sided, maxcopper ground plane, plated through, solder resist, yellow silk-screen. Four 3.3 mm mounting holes.

APPLICATION - TNCS

This modem is obviously only suitable for a TNC if it6 internal modem can be bypassed, and if it provides for the TTL digital signals:

* TXData	e.g. TNC-2 J4-19
* TXClock (16x bit rate)	J4-11
* RXData	J4-17
* Data Detect ("DCD")	J4- 1
* GND	J4-15
* RXClock (optional)	not used

TAPR TNC-2 based designs do this, typified by the TNC-2, PK-80, MFJ1270, TNC-200. Close relatives (but with minor variations) are Tiny-Z, Euro-TNC2C, BSX-2, PK-87, F'K-88 and TNC-220. The necessary interface is at the "Modem Disconnect Jack".

The modem's use is not confined to TNCs, however. Some of the recent multiport packet switches, indeed any digital signalling system, is suitable if it can service the minimum signal set above.

APPLICATION - RADIOS

The ideal would be to have a flat DC-8 kHz radio link. The "better" the TX and RX specification, the better the received data at the detector, and hence less suceptibility to errors.

Some apparently horrid receiver responses still offer useable service, but with a typically 3 db reduction in performance. A good radio achieve6 about 1.5 db implementation loss (compared with a perfect link).

Remember that one is pushing most radio6 to their limit since they were designed for speech where even 100% distortion is still intelligible. A little more finesse is required for data transmission.

RECEIVERS

* NBFM design
* Output from discriminator (essential)
* Response to DC (essential)
* Response no worse than -4 db at 4.8 kHz
* No worse than -10 db at 7.2 kHz
* As smooth/flat a phase delay a6 possible
* As smooth an amplitude response as possible
* Little change in response with 2 kHz de-tuning off-channel
* Symetric, linear FM discriminator characteristic

On the whole, most receivers will perform as required. Those with the least complicated IF filtering appear best, especially with type "D" 20 kHz channel filters (e.g. CFM455D), though the "E" (16 kHz) is OK too.

Radios with dozens of cascaded tuned circuit IFs tend to be fussy, and should be carefully aligned for even response, decent linearity, phase delay and mistuning performance.

TRANSMITTERS

* MUST generate true FM
* Response DC to 7.2 kHz flat (essential)

Transmitters based on Xtal oscillator/multipliers are likely to be the most appropriate. (Usually Base stations". So who wants to tie up a multimode radio anyway!)

Transceivers (synthesised or not) that have quite separate oscillator sub-system6 for generating FM and possibly SSB/CW, which is then mixed with a synthesised source to produce the final carrier am OK.

Simpler synthesised FM transmitters, where the varactor modulated oscillator is within the synthesis PLL are generally not useable, as the PLL tracks the modulation, and so you get no LF response. Them are ways around this by modulating the reference xtal too, but ...!

Remember you need true EM, which preferably means a varactor diode pulling the oscillator frequency, NOT phase modulating a tuned circuit.

9600 BAUD MODEM - DESCRIFTION

All the bits and pieces required to interface digital data to a radio are called a "modem", short for modulator/demodulator. These two functions are complementary, and essentially separate even though there may often be shared parts such as clocks and power supplies etc. Figure 1 is the full circuit diagram of production boards,issue 3. (Issue 1 were prototypes, issue2 the beta-test models).

TRANSMITRANDOMISER/SCRAMBLER

Data for transmission is first passed through a randomiser or scrambler. This ensures that there are no long runs of all "1"s or "0"s or repeated patterns. There are several good reasons for doing this. (Ref 2)

One is that the channel is not DC coupled.' It could never be so in an FM system unless one could guarantee both transmitter and receiver were always on frequency and had no drift. As this is virtually impossible to achieve, one simply AC couples the channel, i.e. gives it a response down to a few Hz, and exploits the feature of the randomised data that it has a negligible "DC" component.

Secondly, since the data stream is now randomised, its spectral energy is evenly spread out at all times. Intense spectral lines do not suddenly appear and create sporadic splatter into nearby channels

A third reason is that since the data is guaranteed to have a regular supply of ones and zeros, the receiver's bit clock recovery and demodulation circuits work better.

Not suprisingly a burst of data sounds like a burst of noise, and is quite hard to distinguish by ear from the unsquelched background.

TRANSMIT WAVEFORM GENERATOR

The transmit waveshape(s) are stored in an EPROM. An8 bit shiftregistercontains the most recent bits, which are used to look up a profile for the middle one. Four samples/bit go to make up the profile. In this way the transmitted waveform is synthesised not only from the present bit, but four that preceded it, and four to come.

The 8 bit value output from the EPROM is converted to a voltage by an inexpensive single-rail DAC, and is then analogue low pass filtered to remove harmonics of the clock and associated discrete phenomena. This is variously called "anti-aliasing", "smoothing" or "interpolating". Either way, it "joins up the dots"!

The arrangement as a whole is a "finite impulse response filter" or FIR for short.

The Transmit EPROM is normally a type 27C128 and can hold between 16 eight-bit long FIRs, to one 12-bit long FIR or various combinations. A 27C256 can also be used, offering up to 32 responses. NMOS roms are also suitable.

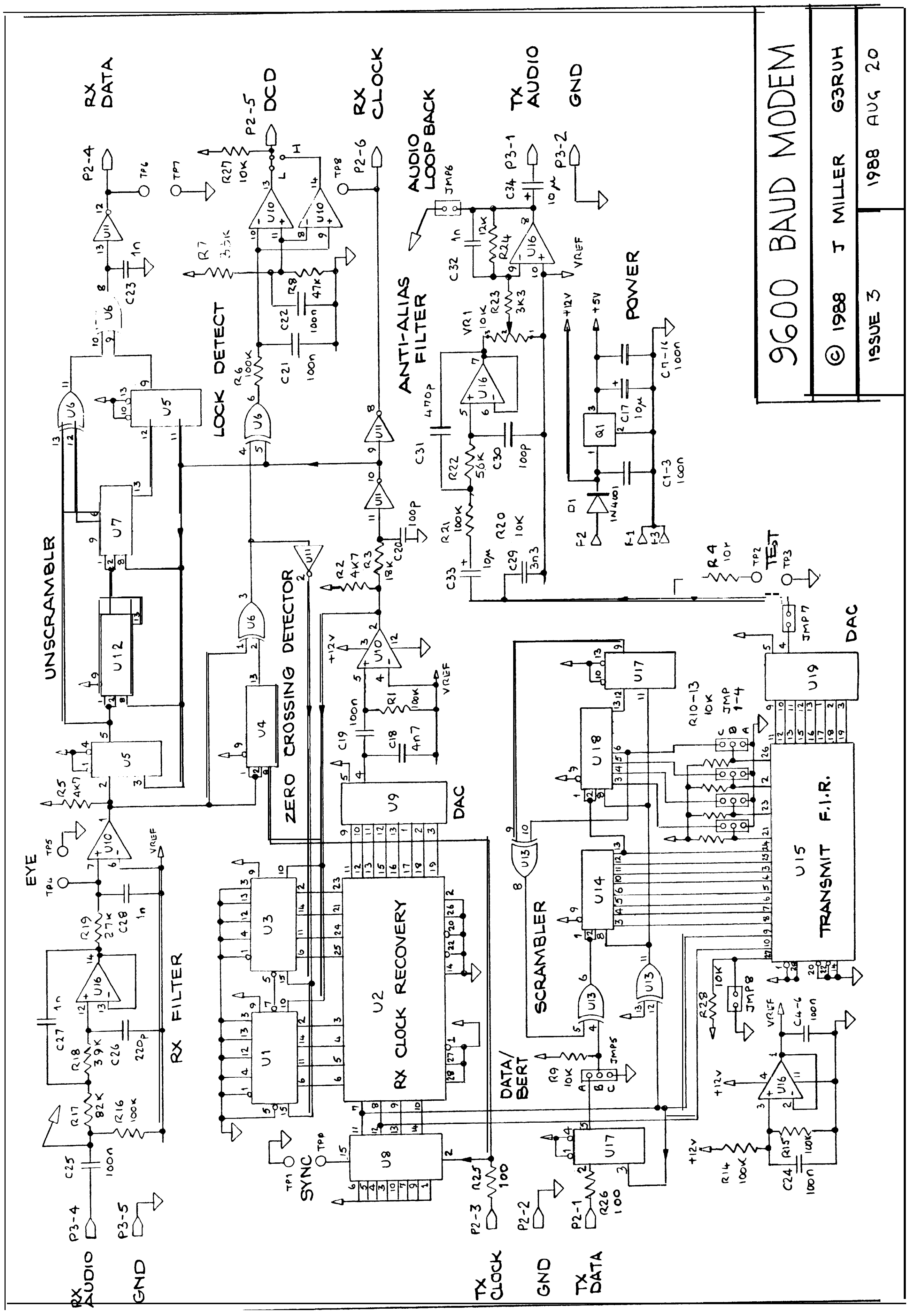
MODEM RECEIVE -FILTER/DETECTOR

Audio from the receiver discriminator is passed through a gentle input filter which removes out of band spurious noise, particularly IF residue. The signal is then limited anddetectedbysampling at the correct instant.

UNSCRAMBLER

The detected data, still randomised is then passed through an unscrambler where the original data is recovered, and this goes off to the TNC. A scrambler is very simple, consisting of a 17 bit shift register and 3 Exor gates. (See ref 2, fig 3).

The scrambling "polynomial" is 1 + X^12 + X^17. This means the currently transmitted bit is the EXOR or the current data bit, plus the bits that were transmitted 12 and 17 bits earlier. Likewise the unscrambling operation simply EXORs the bit received now with that sent 12 and 17 bits earlier. The unscrambler perforce requires 17 bits to synchronise.


9600 BAUD MODEM
© 1988 T MILLER G3RUH
ISSUE 3
1988 AUG 20
RX AUDIO
GND
TX CLOCK
GND
TX DATA
RX DATA
DCD
RX CLOCK
TX AUDIO
GND
RX FILTER
EYE
UNSCRAMBLER
LOCK DETECT
ZERO CROSSING DETECTOR
RX CLOCK RECOVERY
DAC
SYNC
ANTI-ALIAS FILTER
AUDIO LOOP BACK
POWER
SCRAMBLER
DATA/BERT
TRANSMIT F.I.R.
DAC
TEST
P3-4
P3-5
P2-3
P2-2
P2-1
P2-4
P2-5
P2-6
P3-1
P3-2
U1
U2
U3
U4
U5
U6
U7
U8
U9
U10
U11
U12
U13
U14
U15
U16
U17
U18
U19
Q1
D1
1N4001
VREF
+12V
+5V
JMP5
JMP6
JMP7
JMP8
R10-13 10K
JMP 1-4
C1-3 100n
C7-16 100n
C4-6 100n

This polynomial was deliberately chosen to be the same as implemented by Goode (ref 1) in an earlier modem design. It will also be used on one of the UCSAT-C satellite downlinks.

B.E.R.T. TESTING

A particularly useful by-product of scramblers is "bit error rate testing", or BERT for short. Suppose the transmitted data is held to all "1"s. Then a receiver's error-free output should also be all "1"s, even though the transmitted data is quite random. So to test the quality of a link one merely sends all "1"s, and attaches a counter at the other end.

If one bit is corrupted due to channel noise, the error will in fact appear exactly 3 times at the receiver output, because there are 3 versions of the scrambled stream exored together. Even though one error creates two more, this doesn't matter because just the single error is enough for a packet to be rejected.

Incidentally, randomisers/scramblers don't really violate rules concerning codes and ciphers any more than do ASCII, Baudot or Morse. Since the scrambling algorithm is freely published, the meaning of the data is not obscured.

RECEIVE CLOCK RECOVERY

The demodulator must extract a clock from the received audio stream. It's needed to time the receiver functions, including the all-important data detector.

The familiar TAPR TNC-2 state machine is not satisfactory in this application, as its resolution is only 1/16th bit. It can show jitter up to +/- 5/16 ths of a bit in this narrow-band application, which gives bad performance for detector timing.

This modem uses a new digital phase locked loop (DPLL) with a resolution of 1/256th of a bit.

The received audio is limited, and a zero crossing detector circuit generates one cycle of 9600 Hz for each crossing (a proto-clock). This is compared with a locally generated clock in a phase detector based on an up/down counter. The counter increments if one clock is early, decrements otherwise. This count then addresses an Eprom in which 256 potential clock waveforms stored, each differing in phase by 360/256 degrees. In this way the local clock slips rapidly into phase with that of the incoming data.

RX Clock lock-in time depends on the signal to noise ratio, and the initial phase error. A signal that's already in phase pulls into lock in within 0 elapsed bits! A noise free signal exactly out of phase will pull in to the point where data errors cease within approximately 80 bits. A very noisy signal could take up to 200 bits. In practice an average figure is around 50 bits, or 5 ms at 9600 baud.

Proto-clock and local clock are also compared in an exor gate, and when they are "in-phase", a Data Carrier Detected signal (DCD) is sent to the TNC. High or low options are available.

TRANSMIT WAVESHAPE SYNTHESIS

As mentioned, a strength of this modem is its digital generation of the transmit audio waveform. The precise shaping compensates exactly for the amplitude and phase response of the receiver. It also allows very tight control of the transmit audio bandwidth.

The waveform is synthesised as follows. First the "ideal" receiver output waveform (at the "eye point") is defined. This waveform, for one isolated bit, has a perfect "eye". It has a value of +1 at T=0. and a value of 0 at +/-T, +/-2T etc where T is a bit period. Its spectrum is flat to 3300 Hz, -6db down at 4800 Hz, and is absolutely band limited to 6300 Hz. The waveform is called a "Nyquist Pulse".

Next the channel frequency and phase response is measured. It is made up of several parts; the modem transmit anti-alias filter, the radio transmitter response, the receiver response and finally the modem receive filter. In practice most of these components are already characterised, so it's only necessary to measure the radio receiver explicitly.

Now the ideal Nyquist pulse's frequency response is divided by the channel frequency response to give the ideal transmit spectrum. This is then Fourier transformed to the time domain, and specifies EXACTLY the waveform of a transmitted bit that would pass right through the system to emerge with the desired Nyquist shape.

This desired waveform will have a time span of some 15 bits or more duration. However, only the middle part of 8 to 12 bits duration will have significant amplitude. So the extremes are gracefully smoothed off to exactly 8 bits span, a process known as "windowing".

As a verification check, the pulse is now "sent" mathematically forwards through the channel to assay the effect of having to truncate it, and the "eye" point vertical jitter calculated. It is typically +/- 10% of a unit bit amplitude.

This calculation only defines a waveshape for one isolated bit. But it will extend over about 8 bits elapsed time. So the final stage in the synthesia is to add up the impulse responses of all 256 possible combinations of preceding and trailing bits to give the true "convolved" waveform that is finally used. The waveform is then stored as numbers in an Eprom.

The software to do the equalisation is programmed in Basic, except for the Fourier transforms (512 point complex FFTs) which are machine coded for speed. The waveforms and spectra are displayed graphically at every stage. Figure 2 illustrates equalisation of a fairly extreme specimen of radio, a Midland 13-509 220 MHz transceiver (USA).

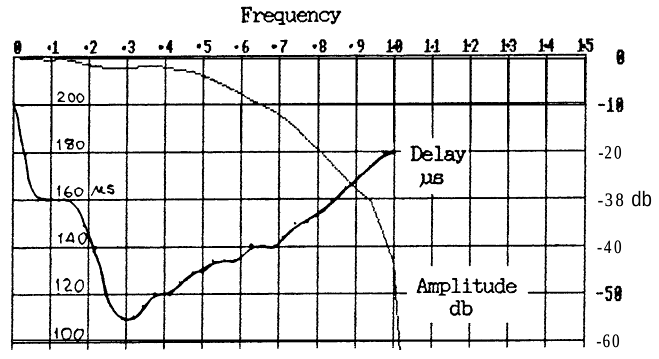


Fig. 2a. Frequency response and phase delay of a typical NBFM receiver. Note that the frequency axis is normalised. f = 1 .0 corresponds to 9600 Hz.

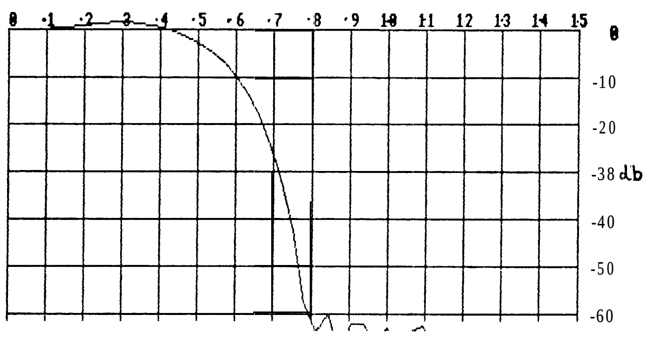


Fig. 2d. Spectrum of the transmitted audio pulse of fig 2c.

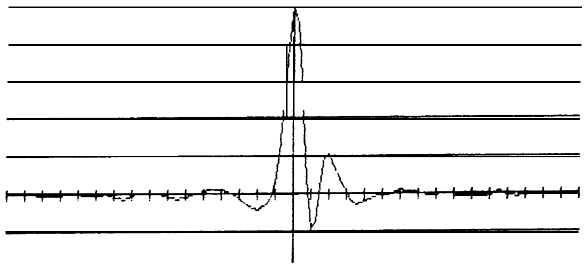


Fig. 2b. Impulse response corresponding to fig 2a. Horizontal axis time "ticks" are at intervals of 1 bit (1/9600 sec), and has been shifted some 125 μs so as to centre the peak. An isolated, unequalised bit would emerge from the receiver shaped like this. A stream of bits would be the sum of many of these. Bad fatures such as significant non-zero values at T = -2, +1 and +2 bits would give rise to substantial inter-bit interference and a very poor eye, making error free communicationimpossible. The purpose of equalisation is to eliminate this phenomenon.

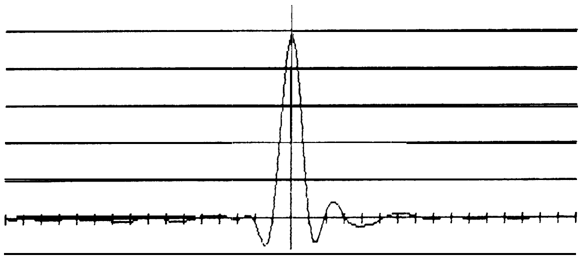


Fig. 2e. Receiver output response to an isolated bit which has been properly equalised. Note that generally zero crossings at the bit ticks (excepting T=0) are zero. This is a "Nyquist Pulse". In fact equalisation is not perfect (e.g. at T=4). This is due to the fairly extreme radio response chosen for illustrative purposes.

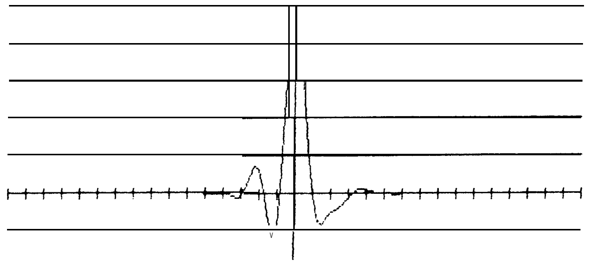


Fig. 2c. Equalised Transmit Bit. If the transmitter sends an isolated bit shaped like this, the receiver will give an output like fig 2e. This is the transmit waveform shaping for one isolated bit. If this corresponds to a binary "1", a binary "0" is a negative pulse like this.

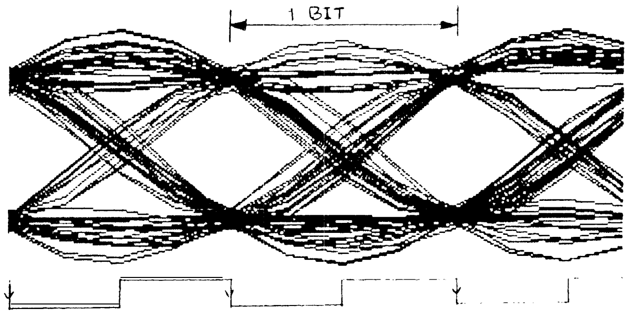


Fig. 2f. The convolution of many bits of fig 2e superimposed looks like this, an "EYE" diagram showing a few hundred bits over 3 bit periods. You would see this on an oscilloscope. The data detector samples this waveform at the widest part of the eye. See how as a result of equalisation the eye is wide open, giving maximum noise immunity. Vertical spread at the sample point is mainly due to the aberration at T = 4 in fig 2e. Note also the considerable spread in zero crossing instants typical of narrow bandwidth systems, which lead to the need for careful clock recovery circuit design.

PERFORMANCE

Users accustomed to the speed of a typical 1200 baud packet channel are usually astonished to see 9600 baud data via radio scrolling off the screen at full speed.

In the perfect environment of audio loopback, the error performance of the modem has been measured, and the implementation loss is about 1 db. In other words the modem itself does not introduce significant degradation in system peformance.

On the other hand it is not very appropriate to describe the modem performance quantitatively in terms of microvolts at an FM receiver input. So much depends on the particular model, and the individual specimen, the environmental noise level, frequency drift, and goodness of equalisation. Steve Goode (ref 1) has given detailed results for one typical situation, and it should be clear from this that there can be no simple snap performance measure, other than "does it work?".

What can be said is that once the received signal strength puts the receiver output above the "spitching" threshold, and into smooth noise, (and that's only a 1 db spread in RF level!) the 9600 baud system becomes essentially error free. So a radio link needs to be just over the noise threshold for good performance. (c.f. 1200 baud AFSK).

Remember, the purpose of this modem is to provide a reliable high speed communications facility - not to scrape weak DX off the noise floor!

EPROM SERVICE

The "standard" transmit Eprom contains waveforms for 16 (or 32) named receivers. One of these is almost certain to be satisfactory for an arbitrary receiver. If not, the author offers a customisation service. Full details are included in the modem Instruction Booklet. New receivers will gradually be added to the standard Eprom as users supply more data.

PCB AVAILABILTY AND SUPPORT

This project is supported with a Printed Circuit Board and full instructions. At the time of writing (88 Aug 16) some 200 are in worldwide use. The board and Eproms may be obtained in various ways:

1. Direct from the author: PCB £18 post paid UK/Europe, £19 airmail USA and elsewhere. TX and RX Eproms (programmed), when ordered at the same time as the PCB, £6/pair. Any eproms ordered separately £5 each. You are welcome to copy the eproms if you wish. Cheque, Eurocheque, Bank draft or cash. Sterling only, no credit cards.

2. From: PacComm Packet Radio Systems Inc, 3652 West Cypress Street, Tampa, FL 33607-4916, USA

3. From: TAPR, PO Box 22888, Tucson, Arizona 85734-2888, USA.

4. From: Siskin Electronics, PO Box 32, Hythe, Southampton, S04 6WQ, England.

ACKNOWLEDGEMENTS

Scores of people have provided feedback and support for this project. These include some 20-25 beta testers world-wide who provided many responses for the eproms, design criticism, and debugged all those wretched non-standard "standard" modem disconnect headers! Gwyn Reedy W1BEL of PacCom Inc and Phil Bridges G6DLJ of Siskin Electronics have been instrumental in promoting the design at both amateur and commercial levels. I also acknowledge the pioneering work of Steve Goode K9NG who did a lot of spade work some years ago, and which convinced me all along that this project was on sound ground. Thanks to Bob McGwier N4HY o.b.o. TAPR for insisting this design should see the light of day and not remain the secret weapon of EastNet-UK's links.

REFERENCES

1. GOODE S. "Modifying The Hamtronics FM-5 For 9600 bps Packet Operation", Proceedings of Fourth ARRL Amateur Radio Computer Networking Conference, pps 45-51

2. HEATHERINGTON D.A. "A 56 Kilobaud RF Modem", Proceedings of Sixth ARRL Amateur Radio Computer Networking Conference, pps 68-75

ARES/Data: A PACKET-RADIO DATABASE FOR EMERGENCY COMMUNICATIONS

W. E. Moemer, WN6I
1003 Belder Drive
San Jose, California 95120
WN6I @ WB6ASR

David Palmer, N6KL
248 Omira Drive
San Jose, California 95123
N6KL @ WB6ASR, CIS: 73357,3157

INTRODUCTION

ARES/Data is a multiple connect, specialized bulletin board system tailored to store and retrieve basic information about people, places, or things during an emergency. The program is a generalized form of the FINDER program (Family Information Database for Emergency Responders), written by David Palmer, N6KL and W. E. Moemer, WN6I[1]. Although ARES/Data allows access to the database via packet radio, the program can also operate stand-alone without the need for packet radio hardware. The actual operating mode is chosen by the system operator when the ARES/Data program is started.

ARES/Data is a system which allows collection and organization of information during a widespread emergency that overloads normal communications channels. The program is designed to be flexible, so that it can be used without change for both small and large disasters to organize information about victims, evacuees, locations, or even ham radio operators. Examples of situations in which ARES/Data could be used are:

- registration of individuals at Red Cross shelters
- patient/victim tracking in a multiple casualty incident
- maintaining staffing information about hams assigned to an emergency
- listings of road closures or damage reports
- logging reports from SKY WARN observers during periods of severe weather

With alternate power sources and their own frequencies, Amateur Radio Emergency Service (ARES) operators can provide the ARES/Data service without tying up critical communications channels or relying on commercial power.

ARES/Data SYSTEM OVERVIEW

There are three major elements to the ARES/Data system:

- ARES/Data software and database
- Data Concentrators
- Voice operators

The central element of the ARES/Data system is the computer on which the ARES/Data program is running. The ARES/Data program collects and collates current information about people or items in the system, according to the needs of the incident. The program establishes and maintains the actual database on floppy disk or hard disk at this central computer. In general, the operator at the computer keyboard can add new records to the database, delete incorrect records, perform searches for specific information, and generate database summaries. The ARES/Data program will run on any IBM® Personal Computer DOS[2] or IBM-compatible system with at least one floppy disk, although a hard disk increases the allowable size of the database and improves performance.

If remote access is desired, addition of a serial port, TNC, and radio allows the central database computer to become the hub of a packet radio network in which up to eight remotely connected stations can access the information in the ARES/Data database. These packet. radio stations, called "Data Concentrators," can update or query the shared database. This data access occurs by exchanging updates or queries in a simple, precise, and well-defined format.

Data Concentrators extend the coverage of the ARES/Data system. They are the input/output ports of the ARES/Data database when remote access is needed. The Data Concentrators can also act as local net controls for any participating voice operators within their range. If voice operators are not needed, the packet operators interact with the public and/or disaster officials directly.

The Voice Operators enter the ARES/Data system when the points of contact with those needing information are numerous and/or spread over a wide area. These amateurs are also the public face of the ARES/Data system. They can be the 'reporters' live at the scene, sending status updates and requests to the Data Concentrators. They also ensure delivery of responses to the persons making status requests.

Emergency responders, their families, evacuees located at a particular shelter, and responsible agency officials access the ARES/Data system by contacting a participating amateur radio operator.

DESCRIPTION and OPERATION of the ARES/Data SYSTEM

THE ARES/Data PROGRAM

The ARES/Data software was written by W. E. Moemer, WN6I, and David Palmer, N6KL. It may be run in either of two modes: stand-alone with no TNC support and no remote access, or by changing the configuration file, the program will control a TNC that allows multiple remote connections. If TNC support is chosen, the program requires a TNC with WA8DED firmware, because host mode is used for communication between the computer and the TNC. No requirement is placed on the other TNCs connected to the ARES/Data database machine, except that they use AX.25 link-layer protocol. The ARES/Data program is written in Turbo Pascal™ Version 4, and uses Turbo Database Toolbox™3 for management and indexing of its B-plus structured tree. Briefly, ARES/Data may be regarded as a specialized multiple connect BBS with a specific command set tailored to the handling of STATUS INPUT information and SEARCH REQUESTS.

The ARES/Data database is simply a collection of records. Each record consists of four main items or “fields” plus a message item. The information in the four main fields can be sorted or searched as required. The rest of this section provides examples and a condensed user manual for the ARES/Data system.

GENERAL RULES FOR CURRENT INFORMATION INPUT / SEARCH REQUESTS

All basic commands can be entered either at the main ARES/Data keyboard or at any one of the remotely connected packet stations. In addition, the operator at the main ARES/Data keyboard (the “sysop”) has an additional set of commands that allow direct communication with the TNC, the printing of a log, backups, and disk report files.

SYNTAX FOR CURRENT INFORMATION INPUT:

To add a record to the database, the operator simply enters values for the four fields and any message, in order, with separators between the fields. The only valid separator is the comma. Within a field, leading and trailing blanks are ignored, but imbedded blanks ARE significant. If no value is desired for a particular field, that field is just skipped by adding an extra comma. The database will fill that field with ten blank characters. For example,

```
field1,field2,field3,field4,
  message<cr>
```

(<cr> means carriage return)

Fields 1 through 4

The four fields are very general. Each can have up to 20 characters, with imbedded blanks. The meaning of each field is defined at the beginning of the event by the ARES officials, depending upon the nature of the event and what type of information needs to be tracked. The sysop can issue a “labels” command that will give specific names to each of the four fields to help the operators remember the purpose of each field.

Message

MESSAGE is an optional, free-form field that can be up to 80 characters in length. It could contain a message, a phone number, an address, or other information deemed useful for the incident.

Examples of Data Input

```
85553195,joe,12,sj34<cr>
Johnson,Mary,93445,sj13,home
     2333  Maplenut  St  SJ
     617-555-2368<cr>
```

All of the input information is stored in the database as a record of the status of a particular person, place, or thing at a particular time and date. The time and date are added automatically by the ARES/Data program. Further STATUS INPUT packets for the same person, place, or thing will also be saved in the database. The time and date identifies which information is most recent.

SYNTAX FOR SEARCH REQUESTS

The search commands instruct the database to look for ALL entries with the same value for field 1, 2, 3, or 4. For example:

`/1,value<cr>`	Searches for “value” in field 1
`/2,value<cr>`	Searches for “value” in field 2
`/3,value<cr>`	Searches for “value” in field 3
`/4,value<cr>`	Searches for “value” in field 4

(For convenience in typing, the question mark "?" may be used instead of the diagonal bar "/"--

both are treated identically). A status report listing all information for each match is sent back to the requesting packet station. The first line gives the search value and the field number. At the end of the report, the line

```
    ARES/Data Search done at HHMM, nn
hits.
```

is sent, which signifies no more information coming, and that "nn" matches (or hits) were found in the database at time "HHMM".

Value

VALUE must exactly match what was originally typed in for the selected field, with leading and trailing blanks removed, and without regard for case.

Examples of Search Requests

```
/1,5553195<cr>
/2,w1aw<cr>
/3,mercyhosp<cr>
/4,85563<cr>
```

SYNTAX FOR SUMMARY REQUESTS

A Summary command is provided that prints a listing of all the distinct entries in a given field, with the total number of like-named items for each distinct entry. For example, if ARES/Data were being used to maintain a list of evacuees, and field 3 was designated for "shelter location", then the command "$3" would print a list of all distinct shelter names in use, and adjacent to each, the number of records (people) in the database at each shelter would be printed.

`$1<cr>`	Produces a summary on field 1
`$2<cr>`	Produces a summary on field 2
`$3<cr>`	Produces a summary on field 3
`$4<cr>`	Produces a summary on field 4

LISTING SPECIFIC ENTRIES (RECORDS) IN THE DATABASE

Each record is automatically assigned a unique record number for identification purposes.

`l nnnnn<cr>`	lists record nnnnn

DELETING SPECIFIC ENTRIES (RECORDS) FROM THE DATABASE

`d nnnnn<cr>`	deletes record nnnnn

This function is always enabled at the sysop keyboard. Its use by remotely connected packet stations is controlled initially by the configuration file during program startup. Thereafter, the sysop can disable or enable this function as necessary. Be extremely careful in using this command! Always list the record first before deleting to be sure you have the right one.

CONFERENCE BRIDGE (Roundtable)

This feature allows any connected station to send messages to other connected stations or to the sysop. The conference bridge illustrates how the ARES/Data system operates as a hub-oriented network, with all transactions passing through the central database station.

Users command:

The users command in the form "users<cr>" or "u<cr>" returns a list of the callsigns of currently logged-on packet stations. The response is of the form:

```
    At WN6I-1:  N6KL  W6BB-3  KJ6K
```

Tell command

The Tell command allows connected packet stations to use ARES/Data as a conference bridge, or roundtable. The general format is:

```
    tell callsign message<cr> or:
    t callsign message<cr>
```

For example:

```
    tell  w6bb-3 We  have  lots  of
people here at SJ12<cr>
```

The message "We have lots of people here at SJ12" is sent to the connected station W6BB-3 prefaced by a time stamp and the call of the station originating the tell command. In this case, if the tell command was sent by KJ6K, W6BB-3 sees:

```
    1230  KJ6K> We  have  lots  of
people here at SJl.2
```

The special callsign "*" or "all" is used to send a message to all connected stations. The special callsign "sysop" sends the message to the sysop at the ARES/Data database station. It is not necessary to enter the entire callsign--just the suffix or some other substring will do. In this case, the message is sent to any connected station whose callsign contains this substring. This feature can be used to create multiple roundtables. For example, packet stations located at, say, hospitals, could adopt sub-station identifiers (SSIDs) of "-1", while those located at shelters could use SSIDs of "-2". This way, broadcast messages of interest to one group can be easily sent without disrupting the other groups. For example:

```
    tell -1 Mercy Hospital has 12
    beds available.<cr>
```

This message would be sent to all stations that were part of the hospital net.

EXAMPLES OF HOW TO USE ARES/Data IN SPECIFIC DISASTER SCENARIOS

- In an evacuation of residents in a local area, the Red Cross often maintains health and welfare status information about evacuees. In this case, the four fields and the comment field might be defined to be:

  Last Name - First Name, Shelter, Number in Family, Last phone, Next of kin

- In a multiple-casualty event where victim transportation needs to be tracked:

  Name, Sex/Age, Ambulance#, Hospital, Injuries

- In a ham radio staffing situation:

  Call, Name, Location, Shift, phone number for cancellation

- In a disaster situation where damage assessment and damage reports are needed:

  Coded type of damage, Location, Number of injuries, Callsign, comment

There are many more possibilities, of course. This is why the exact definitions of the various fields are not defined in advance. In any given situation, more information than will fit into four fields and a comment field might be needed. However, on today's 1200 baud packet radio networks, not much more information per record can be accommodated without restricting the total number of records that can be handled in a reasonable time.

HOW TO OBTAIN YOUR COPY OF THE ARES/Data PROGRAM

The ARES/Data program, a relative of and successor to the FINDER program, is in the public domain. The current version is 0.1, which operates as described in this paper. A copy of the program along with the documentation is available for non-commercial, non-profit use from WN6I or N6KL by sending us a blank, formatted 5 1/4" (360 kB) or 3 1/2" (720 kB) floppy in a mailer with return postage stamps. The cost to you is the cost of the diskette and postage. No other compensation can or will be accepted - please do not send money. We have included a configuration file facility so that you can tailor many parameters to your specific system.

FUTURE DIRECTIONS

The ARES/Data program is continuously being updated to add additional function and flexibility. For example, multiple TNC operation at the main database is being added to the program to allow more data concentrators on multiple frequencies. In addition, support for the Advanced Electronic Applications' PK-232, PK-87, and PK-88 TNCs as the main database TNC is planned. We encourage your comments and suggestions, and will strive to incorporate them in future releases.

ACKNOWLEDGEMENTS

The ARES/Data concept is a generalization of the FINDER system, and the deliberations of the FINDER committee have contributed greatly to the present form of ARES/Data:

Sharon Moemer, N6MWD, FINDER Committee Chairperson
Dick Rawson, N6CMJ
Randy Miltier, N6HMO
Weo Moemer, WN6I
David Palmer, N6KL
Frank Kibbish, WB6MRQ
Bill Robinson, WB6OML
Don De Groot, KA6TGE
Glenn Thomas, WB6W, SCV Section Manager
Don Tsusaki, WW6Z, FINDER manual editor

In addition, we have benefited from all ham radio operators in the Santa Clara Valley Section that have participated in the various alpha and beta tests of the FINDER and ARES/Data systems.

1 *See FINDER: The Family Information Database for Emergency Responders,* Proceedings of the Sixth ARRL Computer Networking Conference, 1987, pp. 134-141, by W. E. Moerner, Sharon Moerner, and David Palmer.

2. IBM is a registered trademark of International Business Machines Corporation.

3. Turbo Pascal and Turbo Database Toolbox are trademarks of Borland International, Inc.

PACSAT SOFTWARE

by Harold Price NK6K and Robert McGwier N4HY

Abstract

The evolving structure of the Microsat system software is discussed. With a launch services agreement in hand, several "Pacsats" should be in orbit in 1989; they will have more memory and a more complex suite of system and application programs than any amateur spacecraft launched to date.

Background

"Pacsat", a dedicated packet radio satellite, is a concept that has been floating around in the amateur digital and space communities for more than six years. The basic concept is simple, using a low orbit satellite to "carry" messages from party to party rather than doing a real-time transfer though a high orbit satellite that both parties can see at the same time. Microsat is a specific AMSAT-NA satellite system that hosts a Pacsat mission.

A paper in these proceedings by Tom Clark gives an overview of the current Microsat design. Another paper by Lyle Johnson describes the hardware that supports the software described in this paper.

One of the large benefits of the amateur radio "Pacsat" concept has always been low cost. Not that hams are necessarily frugal, we'd spend it if we had it. If fact, it has been said that a ham will spare no expense in buying other parts to go with something he got for free. But we haven't got it, so we're forced to be smart. There are several cost reducing concepts in Microsat. One is small size. This reduces building costs (we're talking small, not miniaturized, which drives costs up). It also reduces launch costs, which are typically pegged to launch weight. Another cost reducer is dependence on off-the-shelf commercial grade parts. We're using extended temperature range parts, for the most part, but avoiding the much more expensive space rated parts. A combination of knowing the limitations, a fairly benign low orbit,, and paying attention to the fine details of thermal control, allows us to get away less expensive electronics.

Another cost reducing element is the use of off-the-shelf software, and a low cost software development environment. As anyone who uses computers knows, a low cost software development environment has two components, the actual cost of the software itself, and the cost of the labor required to use the software to do something useful. With Microsat, a serendipitous turn of events has led to a development environment which should be both.

Off the Shelf

Microsat is a very low cost spacecraft. One reason is that it uses off-the-shelf parts and the simplest possible hardware design. The spacecraft itself is minimal, it is "stabilized" only by a bar rnagnet, which locks with the Earth's magnetic field and keeps the spacecraft from presenting the same face to the sun all the time. The interconnecting wiring harness is minimal, using a mere 25 wires and depending on serial bus controllers in each module to multiplex several control and sensor signals on a few wires. The CPU hardware uses standard commercial parts, such as a V40 CPU, closely related to the CPU found in IBM PCs, rather than some exotic, rad hard bit slicing wiz bang.

We'd also like the software to be off the shelf and straight-forward, able to take advantage of standard development tools and languages, and to use large blocks of existing protocol and bulletin board code.

With these goals in mind, Microsat will be the first amateur spacecraft to use an off- the-shelf high level language as its major implementation standard, Microsoft C. It will also use an off-the-s'helf multi-tasking operating system from Quadron Service Corporation called qCF. It should be pointed out that the University of Surrey's UoSat-D, to be launched on the same mission, will also use some of the same software, so the distinction is a shared one.

Why C?

Why, in fact, a high level language? The purpose of the Microsat spacecraft is to support a 16 bit, 5 MHz computer with 10 Mb of memory. In addition to controlling the spacecraft, the computer will support one or more complex data communication protocols, a file system, memory error detection and correction, and data compression. It will compute its sub-satellite point (the point on the ground immediately below the satellite:) so that it can switch the downlink transmitter to low power while over dead zones, or determine when to grab an image from the CCD camera. This will require floating point math. Last but not least, the software must be written in time to support a possible January 1989 launch. Doing all this in assembler would be a daunting task.

C, while not a perfect language, is at least ubiquitous. This means that many people write in it, including many hams. Microsoft C is well known, has copious documentation, has many debugging tools, and is easily available. While it has some bugs, they are well documented, and are in areas we are unlikely to use:, such as graphics. It will generate code that is small enough to fit in the available space, and will run fast enough to support any modem technology we are likely to fly for some tirne to come.

For another advantage, implementations of the amateur radio packet protocols are available in C, including the KA9Q AX.25 and TCP/IP.

C is even nicer though when a support library of interface routines are available to support inter-task communication in a real-time system such as Microsat.

Why Multi-tasking?

Figure 1 shows a number of the tasks that the Microsat CPU will have to support, organized by major function. These tasks will be described in detail later. The chart is greatly simplified, as it leaves out such things as the software to manipulate the 10 Mb of memory, 2 Mb bank switched and 8 Mb as a randomly addressable serial buffer. Many non-related tasks must be carried on concurrently, such as searching the BBS message base for all messages to W1AW while monitoring battery voltage. Some things occur on demand, such as downloading a file, others happen based on time, such as including a telemetry frame in the downlink stream once every n seconds.

One way in which this can be done is to write each function as a subroutine to a single large program and link them all together. The main program would contain a table of subroutines to call when a timer it maintains expires. It would also contain what is commonly called a "commutator loop"; a set of subroutines representing major functions which are called in turn, each does some processing and returns to the main loop. A good example of this type of program is the TCP/IP package by Phil Karn, KA9Q, and others. FO-12 also uses a commutator loop.

Commutator loops have several disadvantages, however. The program is linked into one big unit, meaning that all parts of the program have to be aware of globals and subroutine names, even parts who's only point in common is that they are both called by the same main program Because the program is one linked unit, individual parts can't be easily replaced or reloaded. While commutator-loop programs are effective in the right circumstances, in general the larger they get or the more disparate the parts, the harder they are to develop and maintain. KA9Q is also leaving the commutator behind for a home grown kernal for net.exe for reasons like those given here.

Another alternative to true multi-tasking is a multi-threaded FORTH. The Phase 3 satellites Oscar 10 and 13 use IPS, a FORTH-like system. Much has been said about FORTH and its applicability to real-time control programming, it was originally developed to control large telescopes. With FORTH you either love it or hate it, and most of the prospective programmers for the Microsat project were in the latter category. FORTH is not particularly optimized for things like AX.25 protocols and BBSes, and there are no existing amateur packet radio applications in this language. If FORTH were chosen, we'd be starting far down on the power curve.

A true multi-tasking system would have the following advantages:

1) All programs (tasks) could be writ ten as separate entities. This eliminates global name and other problems, and makes it easier for several widely separated programming groups to contribute for the effort.

2) Since tasks are separate programs, they are easier to generate and test.

3) Tasks can be easily removed and reloaded.

4) The multi-tasking scheduler ensures that all tasks get a chance to use the CPU. More types of programming errors or failures are survivable, a glitch in one task will not necessarily halt processing in other tasks.

A true multi-tasking system seems more desirable, the problem is finding one for a particular hardware system. Fortunately, serendipity smiled on us. Early this year the generic Pacsat CPU design was upgraded from the 1984-style NSC800 Z-80 8 bit class to the 16 bit 8086 class. The serial communications chip was an 8030/8530 surrogate. It just so happens that someone with a long-time interest in PACSAT software (NK6K) was one of the founders of a company that has been marketing a real-time multitasking communication package for an 80186/8030 coprocessor card since 1986. One of the other founders, Wally Linstruth, WA6JPR was a charter member of the ARRL Digital Committee and was one of the first packet users in southern California. In short order, the company, Quadron Service Corporation, agreed to port its software to Microsat, and to give AMSAT free development software and no-cost license agreements for the use of the system on amateur radio spacecraft.

Attributes of the Quadron Multi-tasking system

1) Pre-emptive scheduler. This is a buzzword that means a task is given a certain amount of time to run, and is then placed at the end of a list of tasks waiting to run.

2) Sleep. Tasks can put themselves to sleep, meaning that they have nothing to do at the moment, and don't need to be scheduled on the CPU. Going to sleep is not something that a normal, single task program such as any standard PC-DOS program needs to do. Since there is only one task in such a system, if it has nothing to do it might as well loop. In a multi-tasking system, other tasks may have something to do, and it is more efficient if an idle task is simply not run rather than have it loop until it gets pre-empted. Tasks are automatically put to sleep if they start an operation that can't yet be completed, for example reading an input packet when one hasn't been recieved. The scheduler will start running the task again when input is available.

3) Timers. A sleeping task can also be awoken when a timer it has started goes off.

4) Inter-task messaging. Many of the tasks in Microsat will want to exchange data with other tasks, for example, the BBS will want to send data to the AX.25 output task. This is handled through a mechanism called streams.

A stream is like a little Local Area Network (LAN) connecting tasks together. A task opens to a stream which establishes a name on the "LAN". Other tasks can send messages to that name. Messages are queued. A sleeping task will wake up when a message is sent to it. For example, the AX.25 task sleeps until a task writes a message containing outbound data to it. The AX.25 task processes this data into an AX.25 frame and then writes that frame in a message to the HDLC Driver. The AX.25 handler is also awoken when the HDLC driver sends it a received

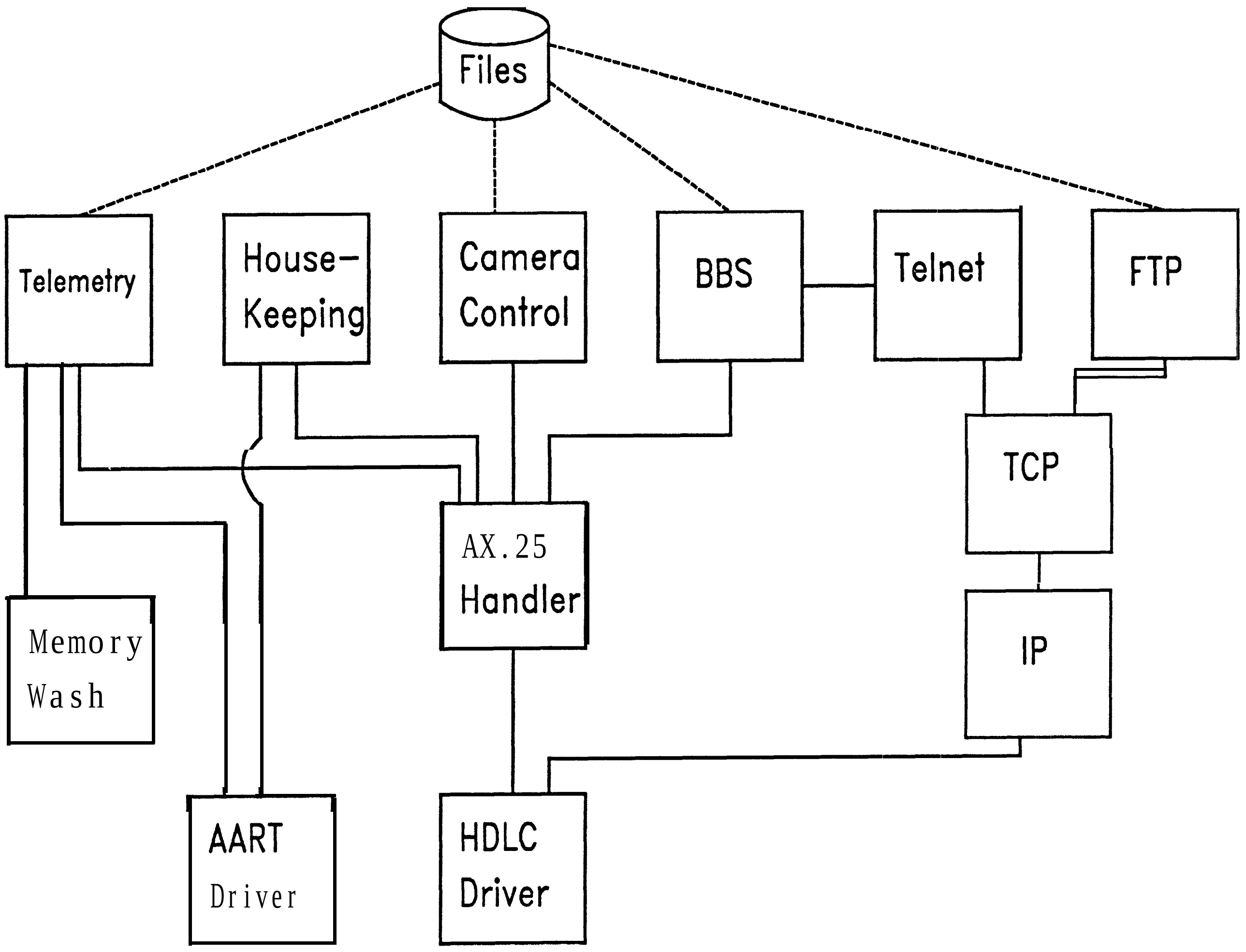


Figure 1. Possible Microsat Task Configuration

irame through a stream. Flow control limits may be set on streams, a task can. be put to sleep if it writes too many messages; it is awoken when the target task reads the messages. This keeps one task from using all the buffers by flooding a second task with messages.

The inter-task messaging is implemented with calls similar to the standard C open, close, read, and write subroutines. Aside from using these calls to send data to other tasks, and the desire to find good places to go to sleep, writing a Microsat application program such as the BBS is pretty much like writing any other C program. Large portions of these programs can be tested on a regular IBM PC using Codeview, which should considerably speed development .

Specifics

Now that we've discussed some of the major design decisions, let's review the major tasks, as shown in figure one. Most of the fine details are still being worked out, the following discussions hit the high points.

Kernal

Not shown in figure one, the kernal supplies the basic multitasking services. It manages the hardware timers, sets up memory, loads and unloads tasks.

File Support

Not shown in figure one, the file support is implemented as a task. The low level C read and write subroutines in the standard C library are replaced by routines that format an I/O request and send it through a stream to the file support task:. Acting much like an IBM PC ram disk driver, the file support task uses the 10 Mb of memory to emulate a disk. File support provides blocking and deblocking services as well as providing error correction for single bit errors (see the Mem Wash description).

BBS

This will be the most visible program to users on the ground. The major goal of the two PACSAT Microsats is to provide a bulletin board and file service. There will probably be two distinct interfaces. One will be an interface familiar to most packet users, the standard RLI/MBL BBS. This is for casual or occasional users who just want to see what's going on or forward an occasional message.

A second interface will be optimized for computer to computer transfers. While the RLI/MBL interface is cur-- rently used for this as well, Microsat will be in a different environment. The current auto-forwarding software is used a network where several hundred stations bang away at each other 24 hours a day, or at least several hours, HF conditions permitting. In the lower latitudes, a Microsat will be visible about 10 minutes at a time, four or five times a day. We'll want to take advantage of every second of that time. We won't want to wait while messages are sent one at a time, reprompting for each new message. We won't want to discard a long message just because the satellite went out of range before the last packet was sent. We'll probably want to block a large number of messages in a single file and send full speed, letting Microsat unblock them later. If a file is partially received, we'd like to be able to continue from the last byte received on the next pass.

The second access method could be used by a smaller number of backbone forwarding stations. The Microsats *will* be an experimental platform for testing various ways of simultaneously maximizing message throughput and maximizing the number of users who directly interact with the spacecraft. On the surface these items appear to be mutually exclusive.

AX.25 handler

If launch occurs as scheduled, the only protocol supported will be AX.25. The software will be a derivative of the KA9Q AX.25 code. It will support a large number of simultaneous connections through all of the uplink channels. When no frames are queued for the downlink, the AX.25 handler will send a message to the Telemetry task asking it to downlink a telemetry record. The telemetry task will also periodically send data based on a timer.

Sometime after launch, (time permitting before), we should also be able to test TCP/IP as an access method in addition to AX.25. AX.25 will at all times remain available.

HDLC driver

The HDLC driver passes frames between other tasks and the uplinks and downlink. The driver is non-trivial. The hardware design supplies several DMA channels, but even so there are more I/O channels than DMA, so the driver must do both DMA and straight interrupt driven I/O. To get the most out of the available processor power, and to enable later Microsat missions to use even higher baud rates, the HDLC driver is written in assembler code. Skip Hansen, WB6YMH, is a real wizard at this sort of thing, and will be porting the HDLC drivers he wrote for Quadron to the Microsat environment.

Although the HDLC driver will probably just be feeding the AX.25 handler at launch, it will later also be the front end for the IP module.

Housekeeping

For all these features to be usable, the spacecraft must be maintained in good operating condition. For example, if the battery voltage goes below a certain preset threshold, low power only should be allowed from the transmitter until recharge. Command stations must be able to talk to the algorithms that monitor optimal settings for power, solar panel operating points, and upload targets for these control algorithms. The command must also be able to manually intervene when the automatic algorithms don't do all that we have asked them to. It might be best to a only allow low power mode over sparesely populated areas on the globe unless a valid packet is heard. These and other spacecraft maintenance functions are handled by the housekeeping modules.

Telemetry

The Telemetry module periodically gathers telemetry data by using the AART driver to collect data from other modules. The data is both sent to the downlink as a UI frame for real-time monitoring, and is also stored in a virtual disk file in memory. The "whole orbit data" format, where the values for telemetry channels are store over several hours and are later downlinked has been proved popular with users by the UO-9 and UO-11 spacecraft. This data would be available in a file and could be downloaded by command stations and users.

The Telemetry module would be addressable via the AX.25 uplink by command stations for the purpose of modifying the interval used for dumping UI telemetry frames and for storing whole orbit data. When diagnosing a problem, the sample rate would be increased, as would the total' memory to be dedicated to storing data. For example, the battery voltage can be sampled once every two milliseconds for a 24 hour period, and the data stored in 8 Mb of memory. It would, of course, take a long time to download that file.

Memory Wash

Some of the memory on the spacecraft is protected with hardware Error Detection and Correction (EDAC) circuitry. When an error is induced in memory by an energetic particle normally filtered out by the atmosphere, the EDAC will correct the error on a read and place the proper data on the bus. The corrected byte is not written back into memory automatically by the hardware. If an error is allowed to linger, there is a chance that a second bit in the same byte will get flipped. Since the hardware can only properly fix single bit errors, you'd like to fix all single bit errors before they become multi-bit. In a process called "washing memory", a task periodically runs through the EDAC memory, reading and writing every byte, causing the corrected byte to be written back into memory over a damaged one.

Most of the memory is not protected by hardware. The reason is economics, 12 bits are used to store each 8 bit byte in hardware protected memory. Hardware EDAC is used for memory that programs run out of, since a program byte in error will usually lead to no good. Software algorithms must be used to protect the remaining memory. This memory is used to store data files and messages. The ramdisk routines will use software EDAC to correct errors, but if a "disk sector" goes unread for too long, multiple bit errors may occur. To reduce this chance, the memory wash task periodically reads all "disk sectors" and writes them out.

The Memory Wash program responds to queries from the Telemetry task and reports the numbers of errors corrected and the current position in the wash cycle. This information then is incorporated into the standard telemetry frame.

AART Driver

This module reads AART commands from other tasks out of the message stream, and sends them to the AART serial control channel. If required, it uses the CPU boards A to D converter to read a data value and return it to the requesting task.

Camera Control

In the CAST Microsat, the primary mission is the CCD camera. In this spacecraft, the BBS will only be used for messages about stored images, and to store the images themselves. Weber State will write this application program.

IP and TCP

As an experiment, the TCP/IP suite of protocols will be ported to the Microsat CPU. This will allow processors such as FTP to be used in lieu of the BBS for file downloading.

FTP

File Transfer Protocol, part of the TCP/IP suite. It will have access to the various data files.

TELNET

The telnet protocol can be used in the TCP/IP suite to pass a stream of characters between a keyboard user and a program, and would thus serve as an alternate path to the BBS.

Implementation Schedule

The software group is aware of the tight schedule for the Spot 2 launch, and the need to balance the desire to do all of the above but be ready for testing in a few weeks. Therefore, we've separated tasks into several groups.

The highest priority group contains those things which must be present if anything useful is to be done with the spacecraft. This group includes the kernal, the HDLC driver, the AX.25 handler, the AART driver. To allow any reasonable development of the applications, these tasks must be present in their near-final form for testing and launch. This group also includes the bootstrap ROM code, which is being implemented by Hugh Pett in Canada, and is beyond the scope of this paper.

The next priority group are the applications which are required to run the spacecraft. Here, very simple temporary modules will do with the fancy ones coming later. The simple ones will be done ASAP; the fancier ones will be done before launch, time permitting. This group includes telemetry and housekeeping and memory wash. It also contains a very rudimentary BBS, with put-message, read-message, and list-message commands. The first BBS will also place its messages in linear memory, not in files.

Next are those things which are required before the final, fancy applications can be written. This is primarily the virtual ramdisk task. Finally comes the additional access methods such as TCP/IP.

Wrap up

This paper has discussed the current thinking in the area of Microsat operating software. Work to implement these plans is underway, some C code has already been run on the wire-wrap CPU prototype. Our choices of high level language, multi-tasking operating system, and implementation methodology have changed the project from impossible to merely difficult.

Acknowledgments

Thanks to Lyle Johnson WA7GXD for another fine hardware design, Chuck Green N0ADI for yeoman work at the wirewrap table; Skip Hansen WB6YMH for his work on the I/O drivers, Hugh Pett for the bootstrap loader; Martin Sweeting G3YJO and Jeff Ward G0/K8KA for past, present, and future collaboration on what a pacsat should do and how it ought to do it. Thanks to Jan King, W3GEY and Gordon Hardeman, KE3D, for stirring up this new project with the initial Microsat idea.

Overview of ARRL Digital Committee Proposals for Enhancing the AX.25 Protocols into Revision 2.1

Eric L. Scace K3NA
10701 Five Forks Road
Frederick MD 21701
USA

0. Summary

A working group within the ARRL Digital Committee has been evaluating enhancements and other proposals for improving AX.25. This paper summarizes the most significant items which are being proposed. You are invited to comment on these proposals. Comments are desired so that any final recommendations by the Committee will benefit from the broadest possible input.

The main topics of discussion have included:

a) improving the channel utilization on busy radio frequencies (e.g., the typical 2 meter simplex channel).
b) cleaning up of minor bugs or ambiguities in the Revision 2.0 specification.
c) suppressing tendencies for connections, under poor conditions, to reappear after disconnect procedures were executed ("Night of the Living Connection").
d) providing support for longer radio callsigns, such as FG/KA3EEN/FS7.
e) providing a mechanism for parameter negotiation between two consenting stations.
f) providing for larger maximum frame sizes, when appropriate.
g) reducing the processing burden for TNCs serving higher speed links (e.g, multiple 64 kbit/s links in a network backbone environment), providing more effective operation on HF radio circuits, and providing better support for information broadcasting applications such as DX spotting networks and QST bulletins.

A key objective of the Committee in considering all of the proposals received was to avoid impact on existing implementations. Every change accepted by the Committee is "backwards compatible"; i.e., an implementation with the feat described below can also successfully connect to, receive connections from, and exchange data with existing AX.25 Revision 2.0 implementations.

1. Improving Channel Utilization

Several mechanisms have been included to improve performance on busy channels.

Round trip timing is measured on a per-connection basis. When frames are transmitted for which an acknowledgement is expected, the time between transmission of the frame and receipt of the acknowledgment is measured. A smoothed average is calculated, using the algorithms described by Phil Karn [correspondence dated 87 Jun 15]. The retry timer value T1, on the first attempt, is set at twice the smoothed round trip time. Details are provided in the companion paper describing the AX.25 data link machine.

Subsequent retransmissions of the same information employ longer Tl values; i.e., larger multiples of the smoothed round trip time. This allows for situations where the channel has become busier. When round trip times improved, the smoothing algorithm automatically begins reducing the retry timer value. Details are found in the companion paper describing the AX.25 data link machine.

AX.25 data link timers are now suspended when a simplex channel is occupied, and timing resumes when the channel becomes idle. This prevents retry timers from expiring during busy periods and triggering retransmissions, when the difficulty is that the remote station just did not have an opportunity to transmit its acknowledgement. Details are provided in the companion paper describing the AX.25 data link machine.

Algorithms for handling multiple simultaneous links in a station are also included. These algorithms provide a round-robin rotation through each data link with traffic to send, prevent any one link transmission from hogging a channel, and relinquish the channel for re-contention after transmission to a particular station. The objective is to give equal opportunity for all links from all stations to use the channel. Details are provided in the companion papers describing the link multiplexor function and the simplex physical channel handler.

For simplex channels, a p-persistence algorithm has been added [Chapponis & Karn, 6th Computer Networking Conference]. This algorithm permits the channel to become more heavily loaded with users without congestion collapse. Details are provided in the companion paper describing the simplex physical channel handler.

2. Cleaning Up Bugs and Ambiguities

Various minor bugs and ambiguities have been clarified in the prose description. The revised prose description has been prepared by Terry Fox, and should be found elsewheres in these Proceedings.

In addition, the Committee has reviewed an extended finite state machine description of the AX.25 protocol. This description follows the graphic conventions of the 2.100 series of Recommendations of the International Telegraph and Telephone Consultative Committee (CCITT), known as System Description Language (SDL). SDL was developed specifically for describing telecommunications protocols. The Committee now proposed to completely replace the present state tables (recognized to be incomplete in some areas, and containing some errors in others) with the SDL descriptions. The SDL descriptions are provided in a series of companion papers.

3. Suppressing the Connection Which Never Dies

During review of the AX.25 SDL diagrams, adjustments were made to disconnect a data link connection under certain error conditions. The previous prose and state table descriptions, as well as implementation decisions made by some designers, had caused disconnected links to unexpectedly reconnect. Unexpected reconnections usually appeared on margin al channels suffering frequent time-outs. These adjustments, plus the improvements described in **§1 above, are** expected to eliminate this minor problem.

4. Longer Callsigns

By far, the most difficult challenge facing the Committee was to find a solution for handling longer callsigns in AX.25 frames which would be compatible with existing implementations. The principle of backward compatibility imposed considerable constraints on this solution which were almost impossible to overcome.

The problem arises when AX.25 is employed in non-amateur radio situations, where callsigns (or “tactical identifiers”) of more than six characters are encountered; e.g., “HQ-WASHINGTON” and "KSGY1497". The problem also arises when AX.25 is used by amateur radio stations operating under certain reciprocal agreements which require: a longer callsign to be used as the legal identification of the station; e.g., FG/KA3EEN/FS7, to pick an extreme case.

The resulting proposal solves the problem by hiding the additional characters of the callsign as a fake digipeater field. Clever positioning of this extension field allows the frames to be digipeated by existing AX.25 implementations. Furthermore, the extension field is *only* included when a callsign is longer than six characters; this means that the vast majority of frames will continue to use address fields of the same form as seen today.

The Committee was unable to achieve 100% backward compatibility with existing implementations in one area. If a “new” implementation attempts to connect to an “old” implementation (or vice versus), the “old” implementation will not properly handle the callsign extension. The workaround is for the “new” implementation to fallback into the existing callsign format when it has been unsuccessfully attempting to complete a connection. The fallback will truncate the callsign to the, first six characters.

This fallback limitation was felt to be acceptable, considering:

a) the vast majority of situations are already handled with 6-character callsign fields, even in non-amateur environments.

b) channels using “tactical” addresses (which tend to be longer) are usually a more controlled environment where the network implementor can equip all stations with “new” implementations.

c) digipeaters are unaffected.

d) the extension mechanism automatically kicks into play only when required; e.g., when the amateur radio operator travels and operates under a reciprocal operating permit which requires the addition of callsign prefixes or suffixes.

e) if an incompatibility between implementations is detected (by a failure to connect successfully), a graceful and automatic fallback to a compatible mode has been provided.

The details of the extension mechanism have been prepared by Terry Fox, and should be found elsewheres in these Proceedings.

5. Parameter Negotiation

Various requests for automatic negotiation of data link parameters have been made. The Committee is proposing to include a negotiation mechanism, based on the HDLC XID frame plus CCITT Q.913 l-style parameter formatting within that frame. Negotiation will only be available between stations operating with “new” implementations; the XID frame will be ignored by existing implementations while the data link connection is in the disconnected state. To maintain backwards compatibility, XID frames will be sent only in the disconnected state; i.e., negotiation occurs automatically and prior to link connection establishment (before SABM).

Presently, the proposal includes negotiation or notification of the following parameters:

a) N1, maximum size of the information field within a frame; but see also the discussion of frame size in 96 below.

b) initial value of round trip timers.

c) transmission speed; this allows an increase (or decrease, if conditions degrade) in transmission speed to occur automatically between compatible stations.

d) use of segmentation procedures; see §6 below.

e) window size.

It was also agreed *not* to include a field for manufacturer proprietary operating modes. It was felt that such proprietary operating modes would potentially segment the TNC population into incompatible subgroups. Such subgroups could not only prevent communications between various implementations, but also potentially interfer with the communications within another subgroup on shared radio channels.

Due to the tight time schedule between the Committee's working group meeting and the publication deadline for these Proceedings, it was not possible to complete a paper detailing the entire proposal. I hope to be able to provide the entire proposal as a handout at the Conference.

6. Larger Frame Sizes

The support of larger frame sizes, like the callsign extension problem, carries backward compatibility difficulties. Existing AX.25 implementations which perform a digipeating function will not support larger frames, even if the source and destination stations are both prepared to accept them.

Therefore, the Committee is proposing to retain the existing N1 value at the 256 octet limit. This limit applies to the information field within I and UI frames; the flags, address fields, frame check sequence, and O-bits added for transparency are not included in calculating the limit.

However, the Committee is also proposing to permit larger N1 values between "consenting stations"; all stations in the connection, including digipeaters, would need to be configured for the larger value. The parameter negotiation procedure discussed in §5 above is one way to obtain "consent".

An additional mechanism is proposed to support applications desiring to transfer larger units of data, while living within the present constraints of N1. This mechanism is a segmentation procedure. The transmitting procedure accepts the large data unit from the application and segments it into multiple smaller frames (I or UI) for transmission by AX.25. The receiving procedure accumulates segments together and then delivers the reassembled large data unit to the destination application. A few octets of overhead are added to maintain segment integrity. The transmitting segmentor alerts the receiver as to the total number of segments to be transferred, and then transfers all segments without interruption. This prevents deadly embrace buffer lockouts. Digipeater operation is unaffected.

The exact segmentation procedure has been standardized by the CCITT in Recommendation 4.931. Again, due to the tight time schedule between the Committee's working group meeting and the publication deadline for these Proceedings, it was not possible to complete a paper detailing the entire proposal. I hope to be able to provide the entire proposal as a handout at the Conference.

7. Higher Speed Operation, Other Types of Links, and Packaging

Finally, it was noted that the amount of computing time devoted to frame analysis could be reduced if certain changes in the format and structure of the link frames were made. Computing time constraints become more important when TNCs support multiple link and multiple radio channels at speeds of 56 kbit/s and above.

Such changes would be fundamentally incompatible with existing implementations.

Therefore, the Committee is proposing to package the enhancements described in §§ 1 through 6 above (i.e., not including changes in format to reduce computing time) as a Revision 2.1 for AX.25. Revision 2.1 is felt to be fully backwards compatible with existing Revision 2.0 implementations.

The Committe will be continuing to evaluate new ideas for:

a) a more compute-time efficient data link protocol for higher speed operation (see other papers within these Proceedings);

b) a data link protocol which would be more effective on HF radio circuits than the present AX.25; and,

c) adjuncts to AX.25 which would be more effective for information broadcast applications, such as DX spotting clusters and "QST" bulletin dissemination.

Introducing System Description Language for AX.25 and Related Protocols

Eric L. Scace K3NA
10701 Five Forks Road
Frederick MD 21701
USA

0. Summary

This paper is part of a series of papers which provide extended finite state machine representations for AX.25 and related protocols. The state machines are depicted using state description language (SDL) graphic conventions from the 2.100 series of Recommendations developed by the International Telegraph and Telephone Consultative Committee (CCITT) of the International Telecommunications Union (ITU). An extended finite state machine representation of a communications protocol such as AX.25 avoids the ambiguities associated with prose descriptions. These descriptions also compel the protocol designer to confront many of the error scenarios which arise on a communications path, and simplify the implementor's task of producing correct solutions which will interwork with solutions created by others.

This particular paper introduces the SDL symbols and explains other general aspects of SDL diagrams. It also lays out an organization of extended finite state machines which, together, perform the AX.25 link layer protocol, handle multiple simultaneous links, interact with the radio transmitter and receiver, and accomodate large data units sent by the application.

1. Status of Proposal

The ARRL Digital Committee proposes to completely replace the existing AX.25 Revision 2.0 state tables with the following SDL representations. The scope of the SDL embraces not only the AX.25 data link procedure used between two stations (and the intervening digipeaters), but also includes descriptions of related protocol functions:

a) segmentation and reassembly of large data units which exceed the N1 limit of a data link.
b) multiple simultaneous links on a single physical channel.
c) contention resolution for shared physical channels.
d) manipulation of the physical transmitter and receiver.

The following material is still in a draft state. You are invited to review and comment on this material. Comments are desired so that the final publication is as useful as possible to its readers.

2. Principles of Extended Finite State Machines

An extended finite state machine models the operation of one party on a communications channel. The condition of the machine is described by its state, a resting condition where the machine awaits input signals from either the application or from remote parties on the communications channel.

Whenever an input signal arises, the machine is triggered to execute a series of operations. These operations may include calculations, the generation of signals to the remote parties on the communications channel, and the generation of signals to the application. The sequence of operations concludes with the machine reaching again a resting state (the same state or a different one).

The entire sequence of operations performed by the machine is *atomic*. For modelling purposes, the sequence is considered to occur instantaneously, and can not be interrupted by any further event. The processing of such further events does not begin until the machine has completed the sequence of operations and has reached a resting state.

Signals may be sent between machines within the same equipment, or via the communications channel(s) to machines in other equipments. These signals are often called "primitives".

Extended finite state machines differ from their cousin, the finite state meachine, in three respects relevant to AX.25:

a) they can maintain internal variables, such as flags, sequence counters, and lists.

b) timers may be set. The expiration of a timer at a later time generates an input signal to trigger the machine to execute a specific series of operations. Timers may be stopped before they expire.

c) internal queues may be maintained. The queues are used to retain input signals (or other information) for processing at a later, more appropriate time.

3. Extended Finite State Machine Model for AX.25

The first figure illustrates the organization of extended finite state machines within a single equipment to implement AX.25 Revision 2.1. In accordance with industry conventions (as embodied in, for instance, the Open Systems Interconnection Reference Model), the machines which are closer to the application are shown as "higher layer" machines, whilst those machines closer to the physical channel are shown as "lower layer".

The collection of machines in the figure embraces both the data link layer and the physical layer of the standard Open Systems Interconnection Reference Model.

3.1 Segmentor

The segmentor state machine accepts input from the higher layer AX.25 user. If the unit of data to be sent exceeds the limits of an AX.25 I or UI frame, the segmentor breaks the unit down into smaller segments for transmission. Incoming segments are reassembled for delivery to the higher layer AX.25 user. The segmentor passes all other signals unchanged.

One segmentor exists per data link. Since a single piece of equipment may have multiple data links in operation simultaneously (e.g., to support multiple higher layer applications), there can be multiple, independently operating, segmentors within the equipment.

Due to the tight time schedule between the Committee's last working group meeting and the publication deadline for these Proceedings, it was not possible to complete a paper containing the SDL description and encodings for the Segmentor. I hope to be able to provide that paper as a handout at the Conference.

3.2 Data Link

The data link state machine is the heart of the AX.25 protocol. It provides all logic necessary to set-up and tear-down connections between two stations and to exchange information in a connectionless (i.e., via UI frames) and connection-oriented (i.e., via I frames with recovery procedures) manner.

One data link state machine exists per data link. Since a single piece of equipment may have multiple data links in operation simultaneously (e.g., to support multiple higher layer applications), there can be multiple, independently operating, data link machines within the equipment.

The data link state machine SDL description is contained in a companion paper found elsewhere in these Proceedings.

3.3 Link Multiplexor

The link multiplexor state machine allows one or more data links to share the same physical (radio) channel. It provides the logic necessary to give each data link an opportunity to use the channel, according to the rotation algorithm embedded within the link multiplexor.

One link multiplexor state machine exists per physical channel. If a single piece of equipment has multiple physical channels operating simultaneously, then an independently operating link multiplexor state machine exists for each channel.

The link multiplexor state machine SDL description is contained in a companion paper found elsewehere in these Proceedings.

3.4 Physical

The physical state machine exists at the physical layer of the Open Systems Interconnection Reference Model. The physical state machine manipulates the radio transmitter and receiver.

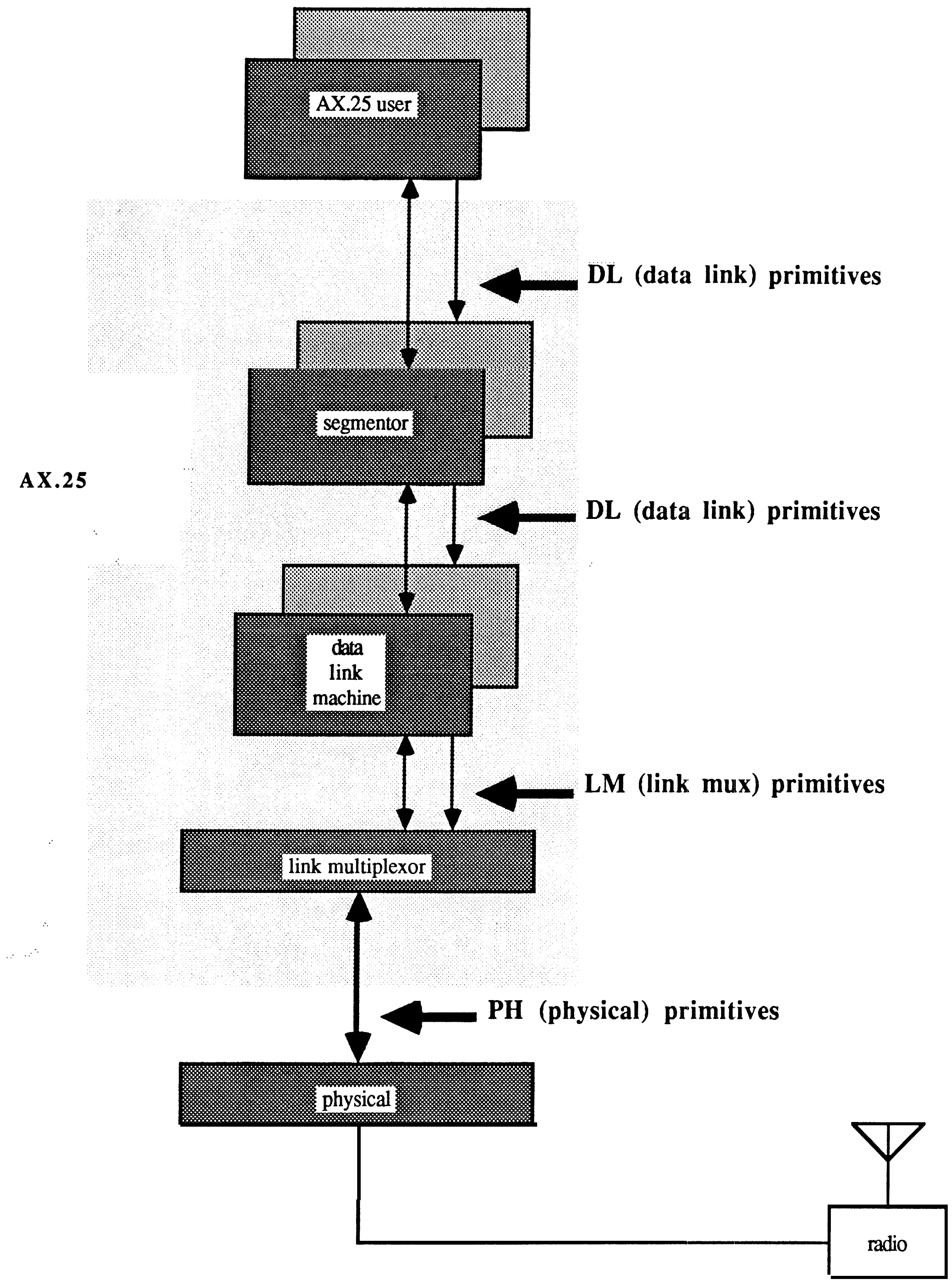
One physical state machine exists per physical channel.

Because different types of radio channel operations are used, the physical state machine comes in different flavors. The intent of each flavor is to hide the peculiar characteristics of each radio channel from the higher layer state machines. Two physical state machines have been defined in SDL:

a) a simplex physical state machine, suitable for use on simplex radio channels (such as 145.01 MHz FM) or on a classical FM repeater. This machine incorporates the CSMA/CD and p-persistence contention mechanisms to permit effective sharing of the radio channel with other stations.

b) a simple full duplex physical state machine. Although full-duplex radio channels are not extensively used to date, the full duplex physical SDL description was developed to illustrate how different flavors of physical channel can work with the other state machines proposed within AX.25 Revision 2.1. This particular full duplex state machine assumes that two stations are assigned a radio channel pair for full-time use (no sharing with other stations), such as would occur between two network backbone switching nodes. Although no sharing is envisioned, intermediate digipeaters and repeaters are allowed

Overview of AX.25 State Machines & Primitives



for in the SDL description.

The simplex physical state machine SDL description is contained in a companion paper found elsewhere in these Proceedings. Time did not permit the completion of the paper describing the full duplex physical state machine, but I hope to have it available as a handout at the Conference.

4. SDL Symbol Definition

The next figure defmes the symbols used in all of the SDL graphic descriptions. You may fmd it helpful to review the text below and the figure along with an actual SDL description from one of the companion papers. The SDL descriptions combine together the operations described by the various symbols into a sequence which is read ***down the page.***

The state symbol denotes the resting states of the extended **finite** state machine. Each state is numbered and named. The sequence number simply indicates the order in which the states are drawn in the SDL. All the permitted sequences of operations **from** a given state originate below the corresponding state symbol. For convenience, each SDL machine is accompanied by **a** summary page which lists, among other things, all of the state names and their corresponding numbers.

Input signal reception (primitive) symbols have notches on either the left or right side. By convention, inputs with the notch on the left are from higher layer (or equal layer) state machines; inputs with the notch on the right are from the lower layer state machine. The name of the input primitive is labeled within the symbol. The SDL machine summary page lists all of the input primitives by name and source.

In addition, the left-notch input signal symbol is used for timer expiration. The number of the expired timer is written inside the symbol. All timers are numbered, by convention, with indications beginning **"T"** and then (usually) a three-digit number. The "hundreds" digit indicates the Open Systems Interconnection Reference Model level number at which the state machine resides; e.g., Tlxx timers are physical layer, **T2xx** are data link layer timers, etc. However, in order to prevent confusion, the present indicators **T1** and T3 are used for AX.25 timers. The SDL machine summary page lists all of the timers by their indicator, and gives a brief description of the purpose of each timer.

Similarly, output signal reception (primitive) symbols have pointers on either the left **or** right side. Output symbols pointing to the left are outputs to the higher layer (or equal layer) state machines; outputs pointing to the right are to the lower layer state machines. The name of the output primitive is written within the symbol. The SDL machine summary page lists all of the output primitives by name and destination.

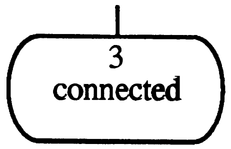
Internal signal symbols are used to post items onto queues (points to left) and to trigger the state machine when something is waiting on the queue to be popped off (notch on left). Each internal signal has **a** description label identifying which queue is involved, and what material is being posted or popped. The SDL machine summary page describes each internal queue used by the state machine.

The save symbol is used to indicate that a particular input event does ***not*** cause operations to be done in the present state. Instead, that particular input event is "saved" until the state machine (triggered by other events) has reached a new and different state.

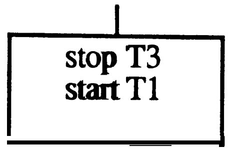
The processing description symbol contains within it a description of internal action(s) executed by the state machine. Examples of these actions are starting and stopping timers, setting and clearing flags, and setting values into variables.

The test symbol is used for branching. The text written within the symbol is posed as a question, and then the appropriate branch is taken.

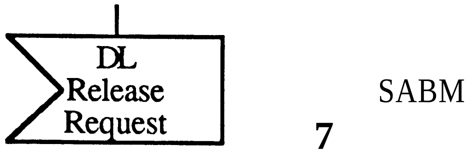
The subroutine symbol is used to encapsulate **frequently** used sequences of steps; the name of the subroutine is written within the symbol. The expansion of the subroutine is listed at the end of the SDL machine description. Subroutine expansions begin with a subroutine start symbol, flow down the page through the specified sequence of operations, and end with the return-from-subroutine symbol. Note that subroutines are not permitted to contain states, nor are they permitted to branch into different return legs. Each subroutine has a single point of return.



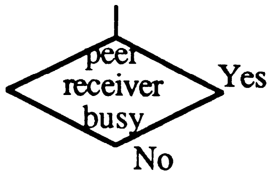
State



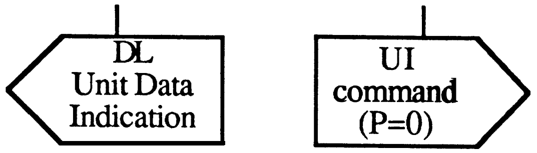
Processing Description



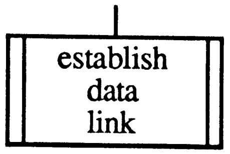
Signal Reception



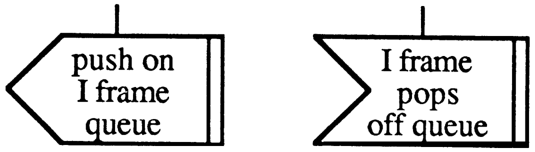
Test



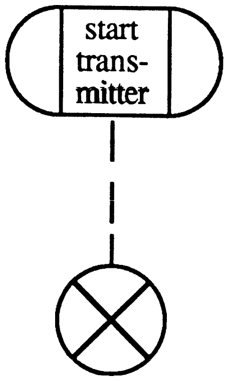
Signal Generation



Subroutine Call

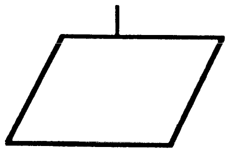


Internal Signal Generation & Reception



Subroutine Start

Return from Subroutine



Save a signal until a new state is reached

AX.25 Data Link State Machine

Eric L. Scace K3NA
10701 Five Forks Road
Frederick MD 21701
USA

0. Summary

This paper is part of a series of papers which provide extended finite state machine representations for AX.25 and related protocols. The state machines are depicted using state description language (SDL) graphic conventions from the 2.100 series of Recommendations developed by the International Telegraph and Telephone Consultative Committee (CCITT) of the International Telecommunications Union (ITU). An extended finite state machine representation of a communications protocol such as AX.25 avoids the ambiguities associated with prose descriptions. These descriptions also compel the protocol designer to confront many of the error scenarios which arise on a communications path, and simplify the implementor's task of producing correct solutions which will interwork with solutions created by others.

This particular paper describes an extended finite state machine which executes the data link procedures between two stations.

1. Status of Proposal

The data link SDL description here is a draft. It borrows heavily from the SDL description of LAPD, developed by the CCITT during 1983-1988 and found in Recommendation Q.921. The data link SDL description was validated by conducting a complete, paragraph-by-paragraph review of the AX.25 prose description and by including annotated cross-references the original working draft. [For simplicity, cross references and other working annotations have not been carried forward into this paper.] The ARRL Digital Committee working group has reviewed this description in detail, and intends to include this machine as an Annex of the upcoming publication of AX.25 Revision 2.1.

Since the working group's review this past July, I have taken the liberty of removing the capability to transmit FRMR frames. This was done for five reasons:

a) According to the present prose description of AX.25, UI frame transmission and reception are prohibited when a station is undergoing FRMR recovery. This seemed to be an unnecessary encumberence to stations which use both connection-oriented and connections data transfer capabilities.

b) During FRMR recovery, the link can not be re-established by the station which has sent the FRMR and is awaiting a mode-setting command.

c) Whilst neither of the preceeding two items are fatal flaws, they do introduce delays in the transfer of information while recovering from very rare error conditions. Recovery from these infrequent errors can be done more simply by just resetting the link with an SABM + UA exchange.

d) An implementation which does not *send* FRMR, but which still receives *and processses* FRMRs, is fully compatible with existing AX.25 Revision 2.0 implementations.

e) Both the SDL description and implementation are simplified. In the SDL description, for example, an entire state is eliminated (plus other ancillary simplifications are made in other states).

Nevertheless, the SDL description to support transmission of FRMR is available, and can be restored to the overall data link SDL machine if desired

The entire data link SDL machine description is still considered to be in a draft state. You are invited to review and comment on this material. Comments are desired so that the final publication is as useful as possible to its readers.

2. Location in the Overall Model

The data link SDL machine forms the heart of the overall AX.25 system. The data link SDL machine resides within the data link layer of the Open Systems Interconnection Reference Model. This SDL machine interacts with the AX.25 user above (through the segmentor SDL machine), and with the link multiplexor SDL machine below.

A data link SDL machine exists for each ongoing communication with a remote station. This communication may use connection-oriented information transfer (via I frames), connectionless information transfer (via UI frames), or both (simultaneously or alternately).

A summary of primitives (signals), flags, error codes, timers, and queues has been compiled on the first page of the SDL diagrams.

2.1 Interaction with the AX.25 User

The AX.25 user directs the operation of the data link SDL machine through the primitives (signals) described below. "DL" in some primitive names stands for "data link".

The DL Establish Request primitive is used by the AX.25 user to request establishment of an AX.25 connection. When the connection has been established (SABM + UA exchanged), the data link SDL machine informs the AX.25 user with a **DL Establish Confirm** primitive. The data link SDL machine uses the **DL Establish Indication** primitive to inform the AX.25 user that connection establishment has occured (triggered by receipt of an SABM from the remote station), and also to inform the AX.25 user that a link reset has occurred.

The DL Unit Data Request primitive can be used by the AX.25 user at any time to transmit information in a connectionless manner via UI frames. The data link SDL machine will deliver any information received via UI frames through a **DL Unit Data Indication** primitive to the AX.25 user.

The AX.25 user submits a **DL Data Request** primitive to the data link SDL machine to cause information to be transmitted in a connection-oriented manner via I frames. The data link SDL machine employs the **DL Data Indication** primitive to deliver information received via I frames.

The Set Own Receiver Busy and **Clear Own Receiver Busy** primitives are sent by the AX.25 user to cause the data link SDL machine to temporarily suspend, and then resume, the flow of incoming information. These primitives affect connection-oriented procedures only. Connectionless information transfer (via UI frames) can not be suspended; if the AX.25 user is not prepared to accept incoming connectionless information, it must discard the information itself.

The data link SDL machine uses the **DL Error Indication** primitive to inform the AX.25 user when frames have been received which are inconsistent with the protocol definition. Error codes are provided with each DL Error Indication to explain the exact type of protocol error which occured.

2.2 Passing Primitives through the Segmentor

The segmentor SDL machine (described elsewheres) passes all primitives through transparently, except for DL Unit Data Request, DL Unit Data Indication, DL Data Request, and DL Data Indication primitives.

The DL Unit Data Request and DL Data Request primitives, received from the AX.25 user, will be segmented into multipled Request primitives (and then sent down to the data link SDL machine) when the amount of data to be transferred exceeds the capabilities of the data link SDL machine (i.e., is greated than N1 octets).

A sequence of DL Unit Data Indication and DL Data Indication primitives, received from the data link SDL machine, will be assembled in a single Indication primitive (and then sent up to the AX.25 user) when the contents indicate that segmentation had been performed.

The details of the segmentor SDL machine are laid out in a separate companion paper. Unfortunately, due to the brief time between the Committee's last working group meeting and the publication deadline for these proceedings, the segmentor paper was not completed. I hope to have it available as a handout at the Conference.

2.3 Interaction with the Link Multiplexor

The data link SDL machine directs the operation of the link multiplexor SDL machine below. "LM" in some primitive names stands for "link multiplexor".

DM, UA, SABM, DISC, UI, I, RR, RNR, and **RE J** primitives are used to convey outgoing frames of these types from the data link SDL machine to the link multiplexor for transmission, and. (with the addition of **FRMR) to** convey incoming received frames of these types from the link multiplexor SDL machine to the data link SDL machine for protocol processing.

The data link SDL machine uses **LM Seize Request** when transmission of an acknowledgement is desired. When the lower layer link multiplexor SDL machine has obtained a transmission opportunity, the multiplexor provides the **LM Seize Confirm** primitive. The data link SDL machine will then generate the correct acknowledgement (e.g., RR, RNR, I frame), as appropriate for that moment, send it to the link multiplexor SDL machine in a primitive, and then conclude with **the LM Release Request** primitive.

3. Overview of States

Unlike the earlier state tables found at the end of the AX.25 Revision 2.0 description, the use of an extended finite state machine allows an effective description to be provided with far fewer states. There are only five states in the **data** link SDL machine.

The Disconnected State (state 0) is the initial state. It is also the state entered when the DISC + **UA** protocol exchange occurs.

The Awaiting Connection State (state 1) is entered when the station transmits **a** SABM command and is waiting for the UA response. When the flag "layer 3 initiated" is set, the **SABM** was transmitted because the AX.25 user had requested connection establishment (or re-establishment); if the flag is clear, the SABM was transmitted as part of an internal data link resetting procedure.

The Connected State (state 3) is reached when the **SABM** + UA protocol exchange has been completed. Transfer of information using **I** frames is now available. Flags are used to implement local and remote busy (Receiver Not Ready) conditions. **A** remote busy condition is periodically revalidated by interrogation of the remote station. An overall limit is imposed on the maximum number of times a remote station can signal busy without acknowledging additional outstanding I frames, to avoid "hangs" caused by infinite **RNRs.** During quiet periods on the link, the remote station is occasionally interrogated to ensure that both parties are still present.

The Timer Recovery State (state 4) is entered when a **T1** timeout has **occured** on a previously-transmitted I **frame,** and the remote station is being interrogated to determine if retransmission is required. Flags continue to implement local and remote busy conditions, as described above.

The Awaiting Release State (state 2) is entered when the station transmits the DISC command **and** is waiting for the UA response.

4. Internal Queue

An internal queue is used for information to be transmitted using the connection-oriented procedures (via I frames).

5. Timers

AX.25 timers **T1** and T3 are implemented with the data link SDL machine. **A** smoothed round trip time (from transmission of **a** frame to receipt of its acknowledgement) is also maintained and used for calculating the appropriate value of TI.

6. Other Parameters Associated with AX.25 Procedures

Present TNC implementations use a variety of parameters in their **AX.25** implementations. Not all of these parameters are found in the data link **SDL** machine. For convenience, many of the popular ones are listed alphabetically below along with their location within the overall family of SDL machines described in this series of papers.

AXDELAY -- time window allowed for an intervening repeater to start operating -- physical (same as **T104).**

AXHANG -- time window allowed for an intervening repeater to stop operating -- physical (same as **T100).**

BEACON -- timer which triggers periodic transmission of a UI **frame** containing a user-specified announcement -- **This** process is considered an **AX.25** user.

CHECK -- timer which triggers interrogation of a quiet AX.25 connection -- data link (same as T3).

DWAIT -- time window allowed for digipeaters to seize a simplex radio channel and begin digipeating -- physical (same as **T101).**

FRACK -- timer supervising the receipt of **acknowledgements** for outstanding I, SABM, and DISC frames -- data link (same as **T1).** Note that this timer has been improved by incorporating a smoothed round trip time calculation, and by setting it to successively larger multiples of the smoothed round trip time as multiple retries occur.

HID -- timer which triggers period transmission of a UI frame containing digipeater identification -- This process is considered an AX.25 user.

MAXFRAME -- maximum number of1 **frames** which can be sent before awaiting acknowledgement -- data link (same as the window size parameter **"k").**

MYALIAS -- **callsign** used by the digipeat function -- link multiplexor (implied in the address check).

MYCALL -- **callsign** used for frames sent and received -- link multiplexor (implied in the address check).

PACLEN -- maximum length of the information field of a UI or I frame -- data link (same as N1).

PACTIME -- a timer which triggers the transmission of a packet containing all untransmitted keyboard input received by the TNC to date -- This timer is considered part of an AX.25 user process.

RESPTIME -- time delay between receipt of a frame before transmission of the response; intended to prevent collision between an acknowledgement and subsequent I frames -- embedded in data link SDL machine, but not explicit. The data link SDL machine instead uses the LM Seize Request and LM Seize Confirm primitives to decide when the opportunity has come: to transmit an acknowledgement. This strategy results in improved acknowledgements, since the acknowledgement sent reflects the exact situation at the time of transmission (rather than the situation at the time that RESPTIME expired).

RETRY -- number of times retransmission is attempted -- data link (same as N2).

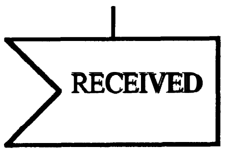
TXDELAY -- time window allowed for remote station to synchronize on incoming flag fill -- physical (same as T105).

UNPROTO -- triggers transmission of a UI frame -- This process is considered an AX.25 user.

DATA LINK

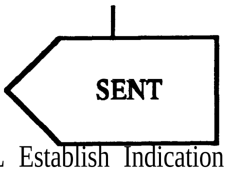
Summary of Primitives, States, Queues, Flags, Errors, and Timers

DL Primitives



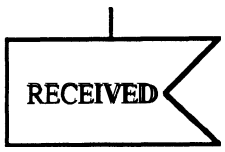
DL Establish Request

DL Release Request
DL Data Request
DL Unit Data Request
Set Own Receiver Busy
Clear Own Receiver Busy



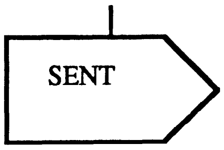
DL Establish Indication
DL Establish Confirm
DL Release Indication
DL Data Indication
DL Unit Data Indication
DL Error Indication

ML Primitives



LM Seize Confirm

DM
UA
SABM
DISC
RR
RNR
REJ
UI
FRMR
I



LM Seize Request
LM Release Request
DM
UA
SABM
DISC
RR
RNR
REJ
UI
I

States

0 -- disconnected.
1 -- awaiting connection.
2 -- awaiting release.
3 -- connected.
4 -- timer recovery

Queues

I frame queue -- queue of information
to be transmitted in I-frames.

Error Codes

A -- **F=1** received but **P=1** not outstanding.
B -- Unexpected DM with **F=1** in states **3, 4,** and 5.
C -- Unexpected UA in states **3, 4,** and 5.
D -- UA received without **F=1** when SABM or DISC was sent with **P=1.**
E -- DM received in states **3, 4,** or 5.
F -- Data link reset; i.e., SABM received in state 3 or 4.
J -- N(R) sequence error.
L -- control field invalid or not implemented.
M -- information field was received in a U- or S-type frame.
N -- length of frame incorrect for frame **type.**
0 -- I-frame exceeded maximum allowed length.
P -- N(s) out of the window.
Q -- UI response received, or UI command with **P=1** received.
R -- UI frame exceeded maximum allowed length.
S -- I response received.

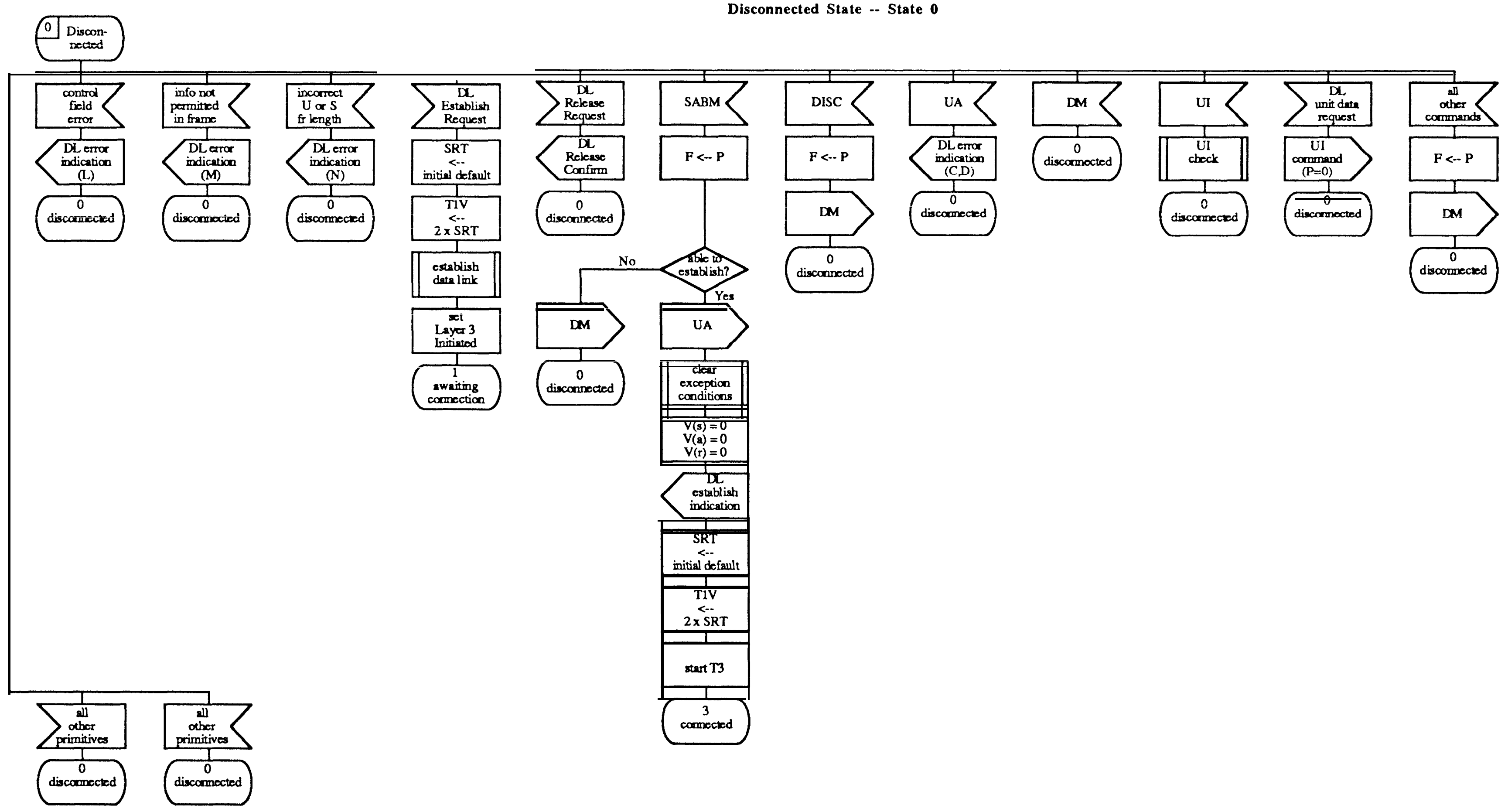
Flags

Layer 3 Initiated -- SABM was sent by request of Layer 3; i.e., DL-Establish-Request primitive.
Peer Receiver Busy -- remote station is busy and can not receive **I-frames.**
Own Receiver Busy -- Layer 3 is busy and can not receive I-frames.
Reject Exception -- a **REJect frame** has been sent to the remote station.
Acknowledge Pending -- **I-frames** have been successfully received but not yet acknowledged to the remote station.

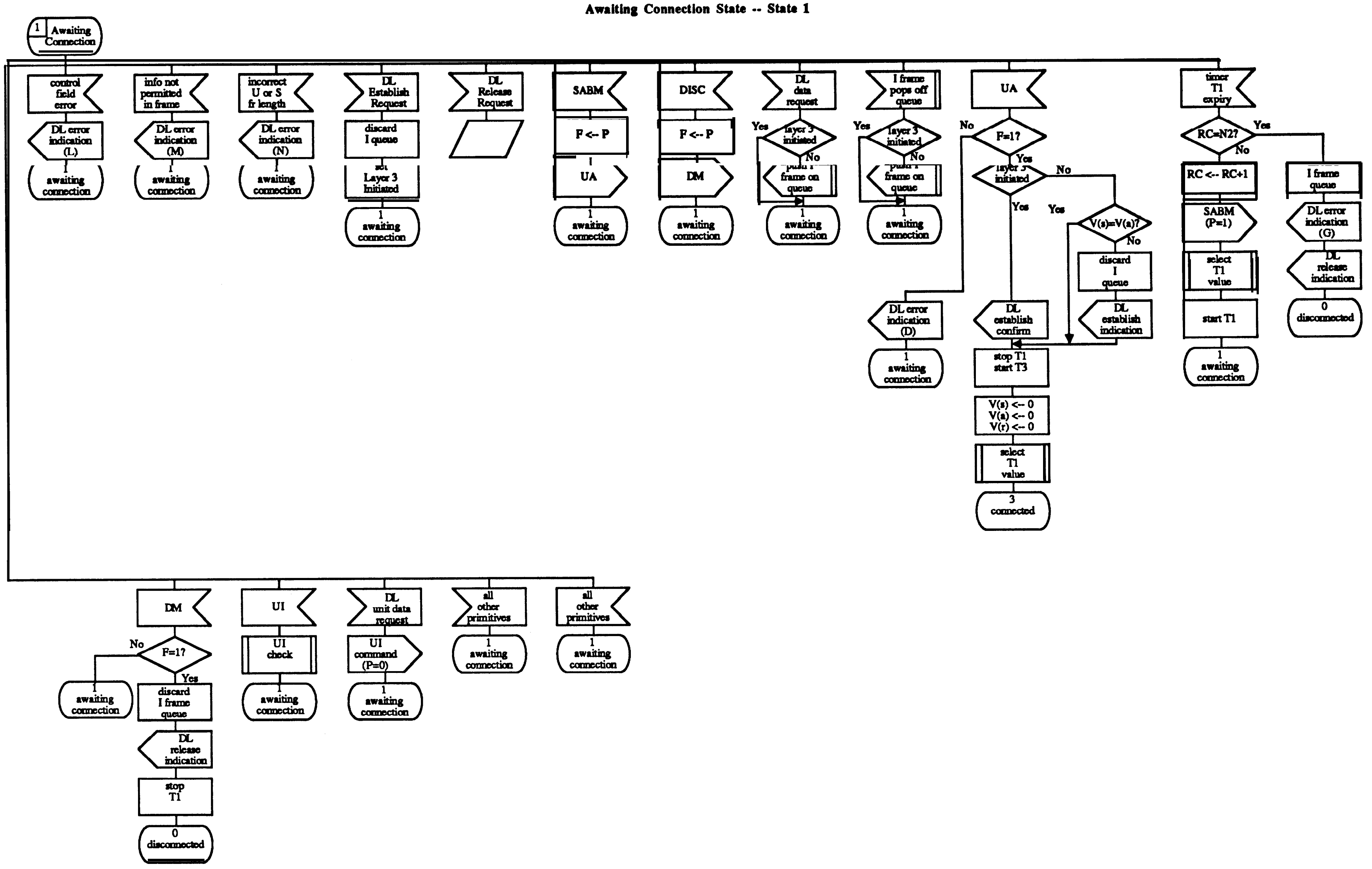
Timers

SRT -- smoothed round trip time.
T1V -- next **T1** value; default initial value is initial value of SRT.
T1 -- outstanding I-frame or P-bit.
T3 -- idle supervision (keep alive).

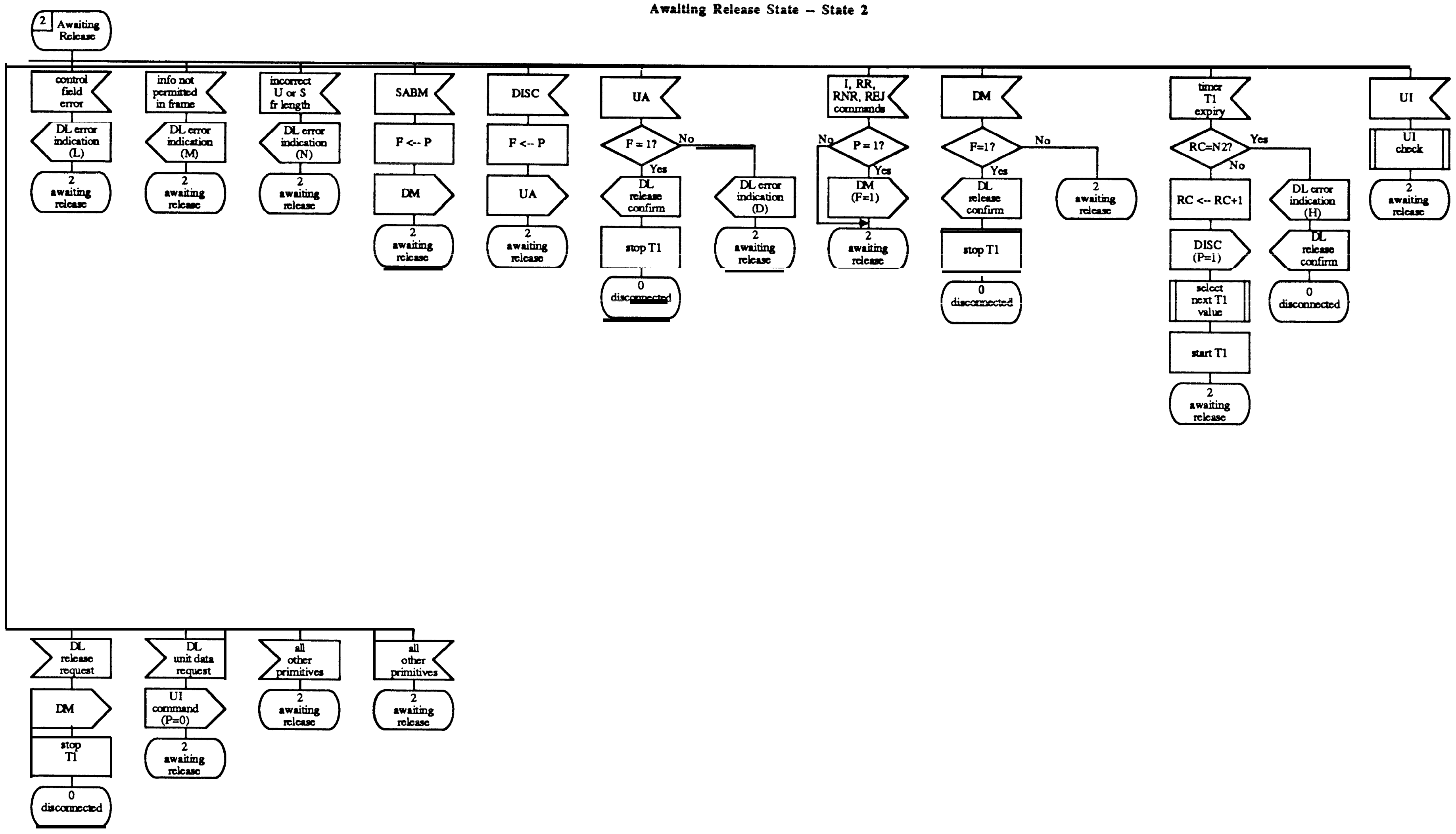
Data Link
Disconnected State -- State 0



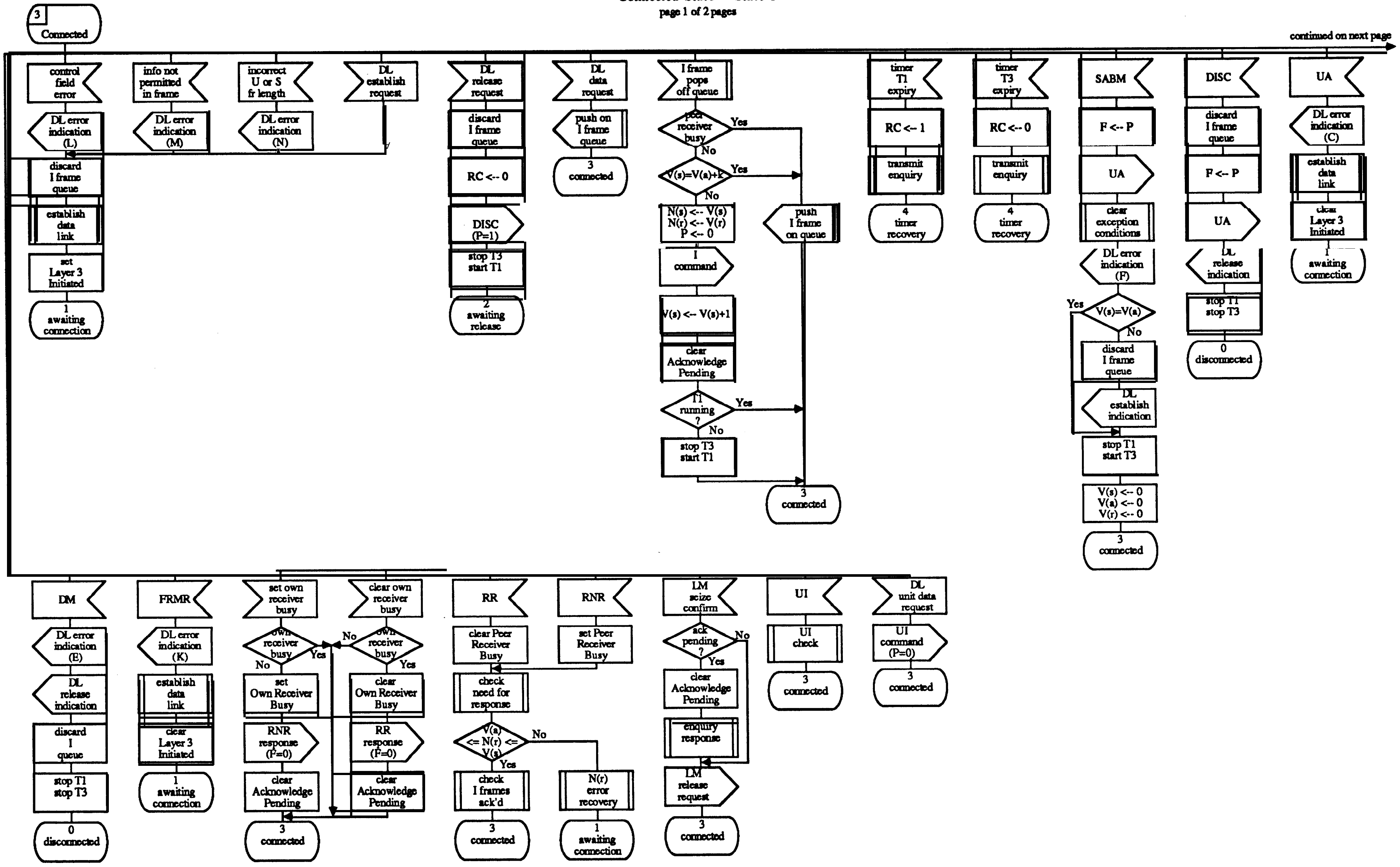
Data Link
Awaiting Connection State -- State 1



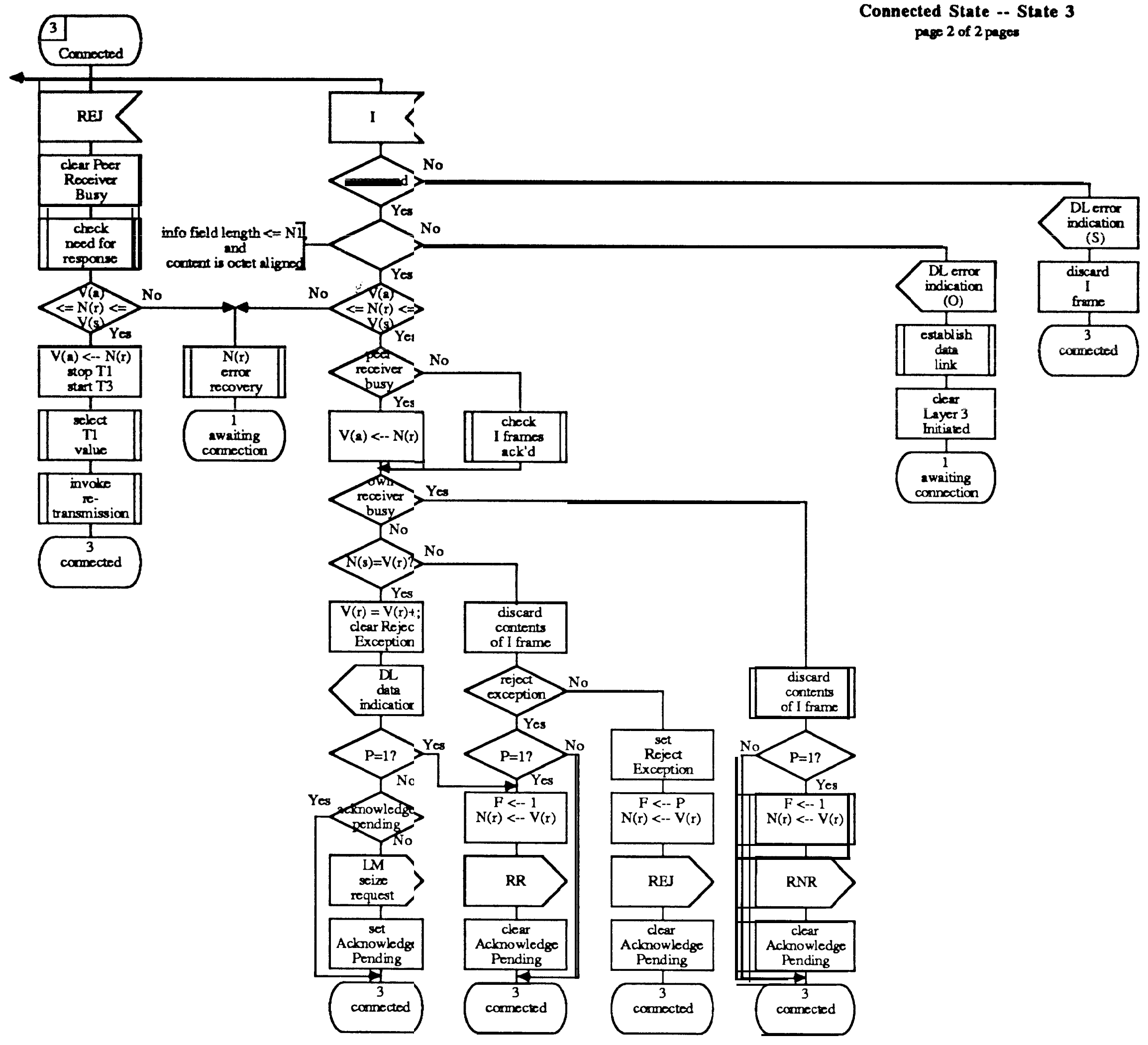
Data Link
Awaiting Release State -- State 2



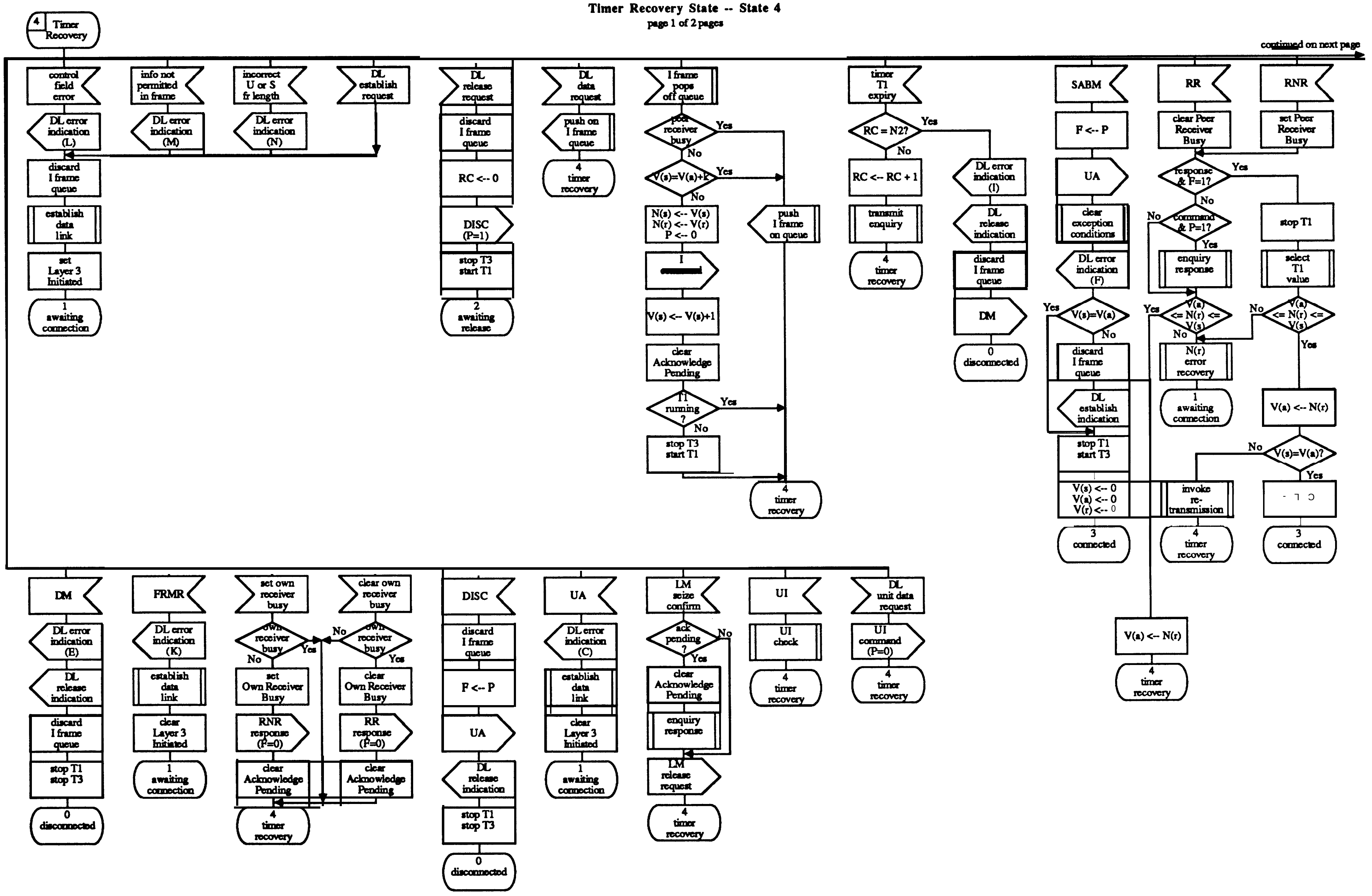
Data Link
Connected State -- State 3
page 1 of 2 pages



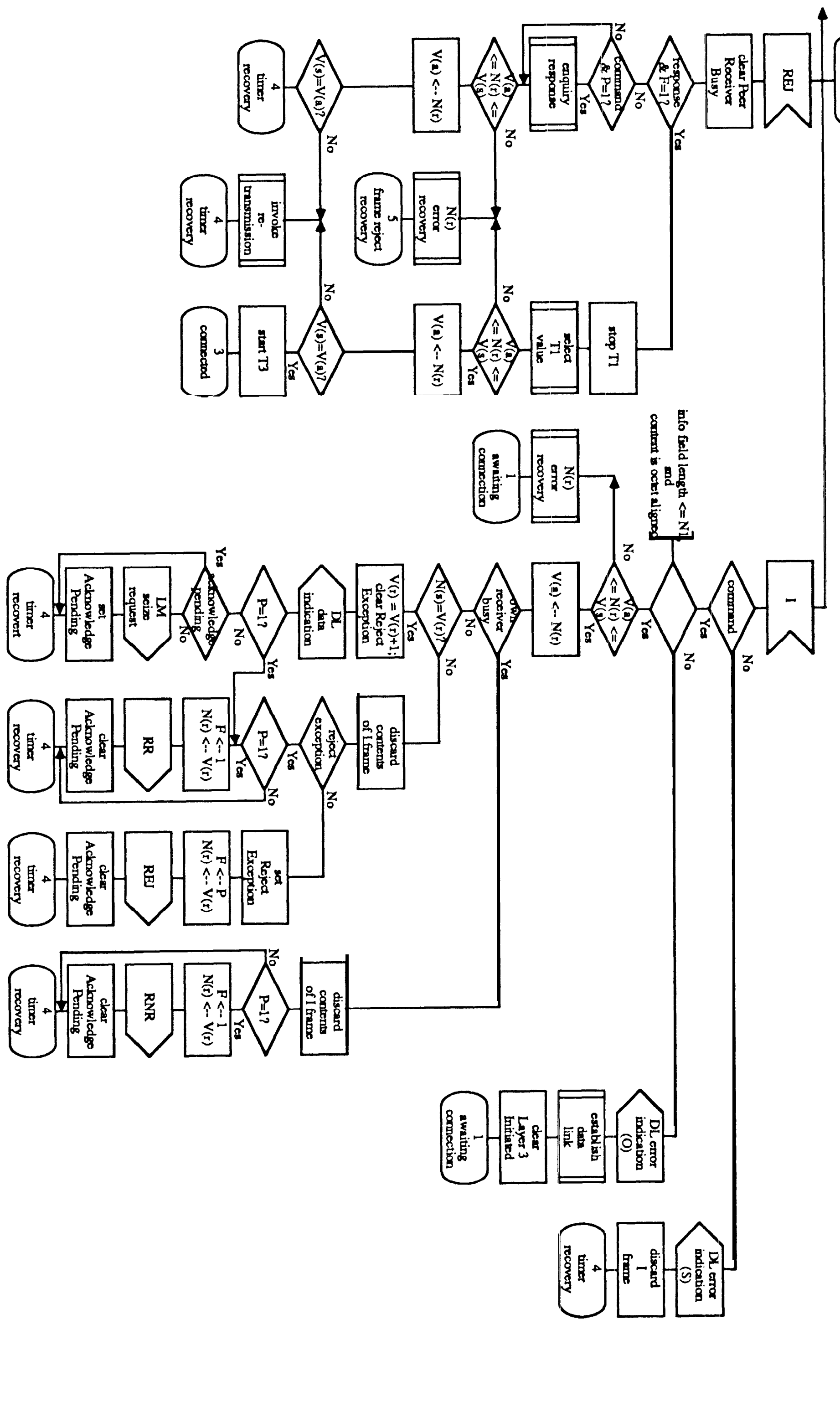
Data Link
Connected State -- State 3
page 2 of 2 pages



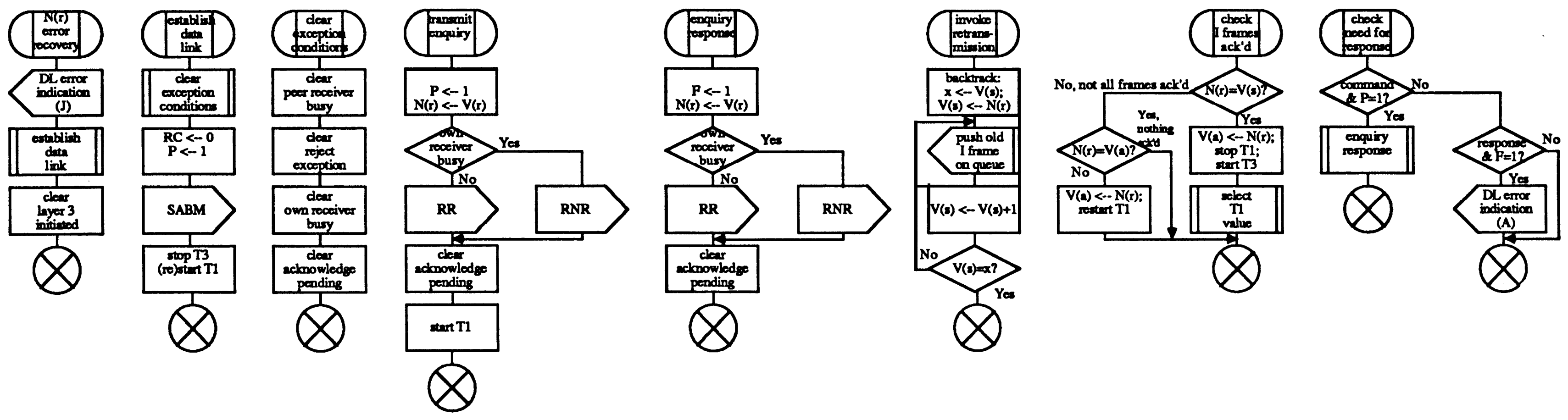
Data Link
Timer Recovery State -- State 4
page 1 of 2 pages

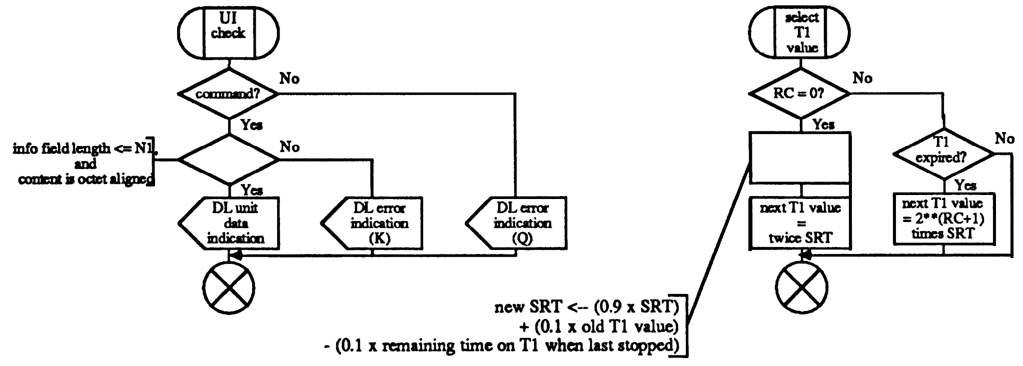


Data Link
Timer Recovery State -- State 4
page 2 of 2 pages



Data Link
Subroutines





AX.25 Link Multiplexor State Machine

Eric L. Scace K3NA
10701 Five Forks Road
Frederick MD 21701
USA

0. Summary

This paper is part of a series of papers which provide extended finite state machine representations for AX.25 and related protocols. The state machines are depicted using state description language (SDL) graphic conventions from the 2.100 series of Recommendations developed by the International Telegraph and Telephone Consultative Committee (CCITT) of the International Telecommunications Union (ITU). An extended finite state machine representation of a communications protocol such as AX.25 avoids the ambiguities associated with prose descriptions. These descriptions also compel1 the protocol designer to confront many of the error scenarios which arise on a communications path, and simplify the implementor's task of producing correct solutions which will interwork with solutions created by others.

This particular paper describes an extended finite state machine which supports multiple simultaneous AX.25 links. "Link" here embraces both AX.25 connections (established between two stations with the SABM command) and connectionless operation between two stations (using UI frames exclusively). The main responsibilities of the link multiplexor SDL machine are to insure that each link has a fair and equal opportunity to access the radio channel, and to handle incoming frames which require digipeating.

1. status of Proposal

The link multiplexor SDL description here is a draft. The ARRL Digital Committee intends to include this machine as an Annex of the upcoming publication of AX.25 Revision 2.1.

The present AX.25 Revision 2.0 includes very little information about the use of simultaneous links. It has been my personal observation that a number of AX.25 implementations which permit simultaneous links fail to operate in a graceful manner. For example, some implementations transmit every outstanding frame, regardless to whom it is destined, in a single (and sometimes quite lengthy) burst... and then sit back and expect that twenty or more remote stations will successfully acknowledge receipt before twenty-plus concurrent retry timers expire. Even under optimum conditions extensive polling and retries result. The proposal for a standard link multiplexor is intended to provide helpful guidance which will lead to more effective implementations of stations which use simultaneous connections.

The following material is still in a draft state. You are invited to review and comment on this material. Comments are desired so that the final publication is as useful as possible to its readers.

2. Features of the Link Multiplexor SDL Machine

The link multiplexor SDL machine includes the following features:

a) preferential treatment for frames to be digipeated.
b) discard of incoming frames containing FCS errors.
c) round-robin rotation for access to the radio channel between all links with frames awaiting transmission.

[Note -- An equivalent link multiplexor SDL machine with priority assignments for each link has also been developed, but is not included here. If there is sufficient interest expressed, it will be made available for inclusion in AX.25 Revision 2.1.]

3. Location in Overall Model

This SDL machine resides within the data link layer of the Open Systems Interconnection reference model. The link multiplexor SDL machine interacts with data link SDL machines above it and a single physical layer SDL machine below.

The physical layer SDL machine may be either the simplex physical SDL machine or the simple full duplex physical SDL machine (as described in companion papers), or any other physical layer SDL machine which is prepared to accept and generate the corresponding primitives.

The link multiplexor SDL machine works with multiple data link SDL machines above it. A data link SDL machine exists for each ongoing communication (connection-oriented or connectionless or both) with a remote station. The individual data link SDL machines are distinguished by hypothetical "identifiers". In formal protocol standardization, these hypothetical identifiers are known as service access points; however, in practical terms for AX.25 the identifier is the **callsign** of the remote station (as found in the address field of the AX.25 **frame).**

3.1 Interaction with the Data Link

The data link SDL machine directs the operation of the link multiplexor SDL machine through the primitives described below. **"LM"** in some primitive names stands for "link multiplexor".

LM Seize Request -- This primitive requests the link **multiplexor** SDL machine to arrange for transmission at the next available opportunity. The data link SDL machine uses this primitive when an acknowledgement must be ma&, but the exact frame in which the **ackowledgement** will be sent will be chosen when the actual time for transmission arrives. The link multiplexor SDL machine uses the **LM Seize** Confirm to indicate that the transmission opportunity has arrived. After the data link SDL machine has provided the acknowledgement, the data link SDL machine gives permission to stop transmission with the **LM Release Request** primitive.

Frame -- This primitive from the data link SDL machine provides an AX.25 frame of any type (UI, SABM, I, etc.) which is to be transmitted. An unlimited number of frames may be provided. The link multiplexor SDL machine accumulates the frames in a **first-in** first-out queue until it is time to transmit them.

During reception, the link multiplexor SDL machine delivers an **incoming** AX.25 **frame to the addressed data link SDL machine in a Frame primitive.**

3.2 Interface to the Physical Layer

The link multiplexor SDL machine works with **a** specific physical **layer SDL machine. The** exact type of physical Iayer **SDL machine** used is dependent on the characteristics of the radio channel. **Two** companion papers described physical layer **SDL machines** suitable for simplex channels typical of most 2 meter amateur packet frequencies, and for simple (used only by two stations) full duplex channels which would be part of a backbone networking trunk circuit. Importantly, the variation in operating characteristics of radio channels is kept hidden from the link multiplexor SDL machine. The link **multplexor** SDL machine uses the same primitives to communicate with the physical layer SDL machine, regardless of which type it is.

PH Seize **Request** is used by the link multiplexor SDL machine before each transmission to request access to the radio channel. When access has been obtained (i.e., the transmitter is operating, any **intervening** repeater has had an opportunity to be activated, the remote station's received has had an **opportunity** to become synchronized, and the channel is considered ready to send traffic), the link multiplexor SDL machine is notified by a **PH Seize Confirm primitive.**

At **this** point **the link multiplexor** SDL machine **delivers** each **frame to be** sent **in a Normal Frame primitive to the** physical layer **SDL machine.** When **all** frames *which have been awaiting transmission for a given link* have been submitted for transmission, the link multiplexor SDL machine concludes with a **PH Release Request** primitive. The intention here is that **a single** transmission **will** contain frames for only one remote station. The PH Release Request primitive permits the physical layer **SDL** machine to release the channel for use by others, for digipeating, and for receipt of acknowledgements in a contention environment (such as shared **simplex** channels).

The physical layer SDL machine provides incoming frames to the link multiplexor SDL machine via **Frame** primitives. The link multiplexor SDL machine checks each incoming frame for FCS errors. Correctly-received frames are checked to see if digipeating by the station has been requested and if the digipeat function is enabled (a user specified parameter); if so, the frame is resubmitted to the physical layer SDL machine **in** a **Digipeat Frame primitive.** (PH Seize Request and PH Release Request are not used for **digipeat** operation.) Correctly-received **frames** addressed to this station are delivered to the indicated higher-layer data link SDL machine (described earlier in. **§** 3.1).

The physical layer SDL machine **also** provides the **PH Busy Indication,** which is used by the link multiplexor SDL machine **to** suspend **all AX.25 data link timers.** Timers resume ticking when **the PH Quiet Indication is** received. The suspension of timers overcomes **a** problem noted in some implementations on busy channels. This problem occurs when frames are transmitted and **a** response is expected. If the channel is busy, it is possible for the retry timers (AX.25 timer **T1)** to expire before the remote station had an opportunity to send any acknowledgement. This premature expiration causes needless retries and polling, further cluttering an already busy frequency.

4. Internal Operation of the Machine

The internal states, queues, and flags are summarized on the first page of the SDL diagram.

All queues are **first-in first-out** queues. Three queues are utilized, in conjunction with two **flags,** to implement round-robin rotation amongst the various data link SDL machines.

The Awaiting Queue contains all primitives received from data link SDL machines which have not yet had an opportunity to transmit.

When a primitive pops off the Awaiting Queue, it and all other primitives from that same data link SDL machine are placed in order on the Current Queue. The identity of this data link SDL machine is maintained in the Current DL flag. The link multiplexor SDL machine then proceeds to obtain a transmission opportunity for that data link SDL machine. Any further primitives received from that particular data link SDL machine are added to the Current Queue. When the transmission opportunity arrives, everything in the Current Queue is conveyed to the physical layer SDL machine for transmission. (In the event of an overly large amount of information to be sent, the physical layer SDL machine makes whatever breaks in transmission are appropriate for reasonable channel sharing. This is done within the physical layer SDL machine and hidden from the higher layers.)

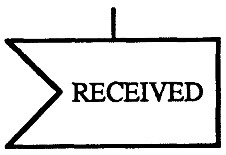
Once everything has been sent for the current data link SDL machine, its identity is moved to the Served List. Any subsequent primitives from this data link SDL machine are added to the ServedQueue.

The link multiplexor SDL machine then goes back to the Awaiting Queue to pop off the next primitive, and thereby identify which data link SDL machine has the next transmission opportunity. If the Awaiting Queue is empty, then the link multiplexor SDL machine concludes that all data link SDL machines which had frames to be sent have now been served. The queue system is reset by converting the Served Queue into a new Awaiting Queue, and by purging all identifiers from the served List.

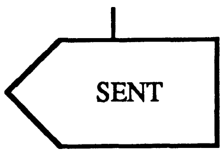
LINK MULTIPLEXOR

Summary of Primitives, States, Flags, Errors, and Timers

LM Primitives

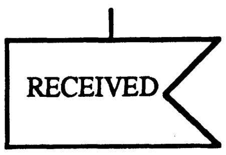


LM Seize Request
LM Release Request
frame (any type)

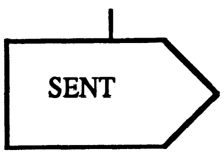


LM Seize Confirm
frame (any type)

PH Primitives



PH Seize Confirm
PH Quiet Indication
PH Busy Indication
frame



PH Seize Request
PH Release Request
digipeat frame
normal **frame**

States

O-Idle
1 -- Seize Pending
2 -- Seized

Error Codes

No error codes used.

Queues

Awaiting Queue -- queue of primitives received from data link machines which are not presently using the transmitter.
Current Queue -- queue of primitives received from the data link machine which is presently using the transmitter.
Served Queue -- queue of primitives received from data link machines which already have used the transmitter.
Note -- After all data link machines have had an opportunity to be served, then the Served Queue is converted to the Awaiting Queue.

Flags

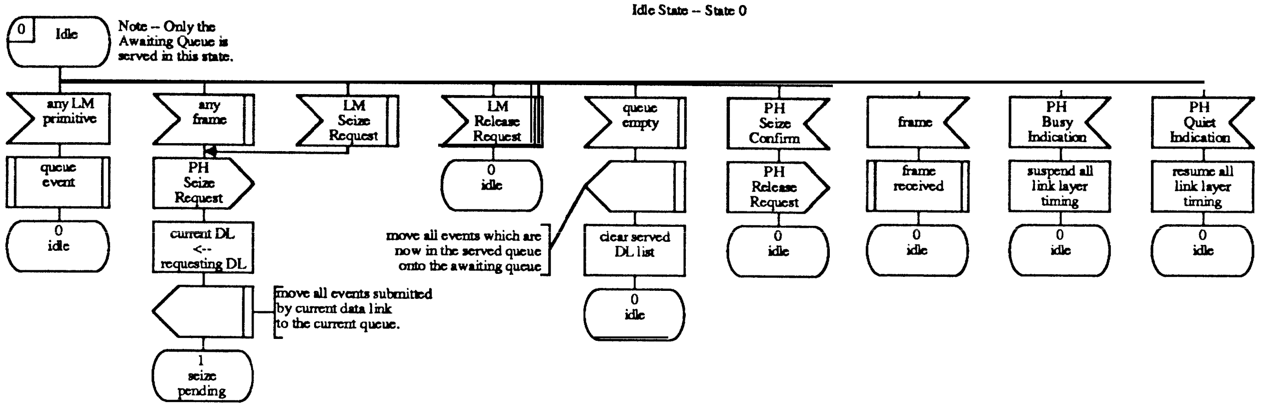
Current DL -- Identifies the data link machine currently using the transmitter.
Served List -- Identifies the data link machines which have already used the transmitter. This list is cleared when all data link machines with **frames** to send have been served.

Timers

No timers used.

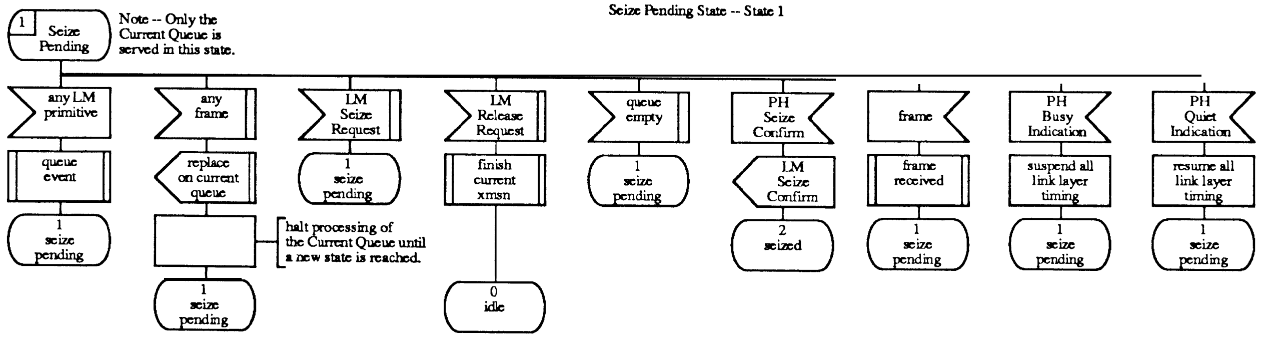
Link Multiplexor

Idle State -- State 0


0 Idle
Note -- Only the Awaiting Queue is served in this state.
any LM primitive
queue event
0 idle
any frame
PH Seize Request
current DL <-- requesting DL
move all events submitted by current data link to the current queue.
1 seize pending
LM Seize Request
LM Release Request
0 idle
move all events which are now in the served queue onto the awaiting queue
queue empty
clear served DL list
0 idle
PH Seize Confirm
PH Release Request
0 idle
frame
frame received
0 idle
PH Busy Indication
suspend all link layer timing
0 idle
PH Quiet Indication
resume all link layer timing
0 idle

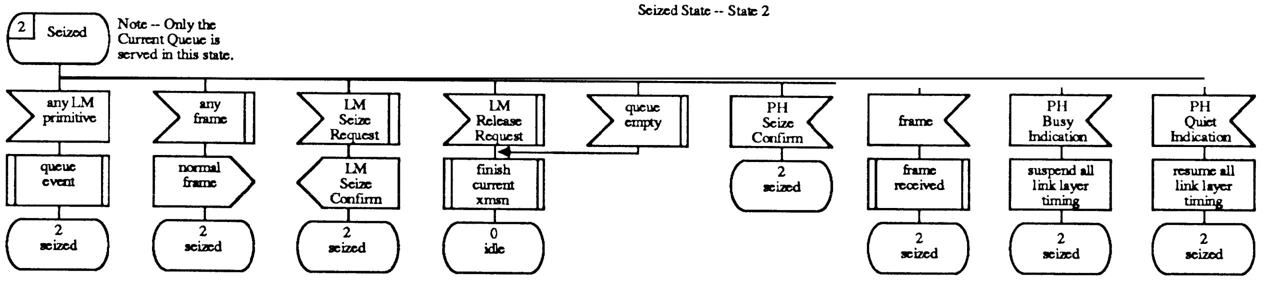
Link Multiplexor

Seize Pending State -- State 1



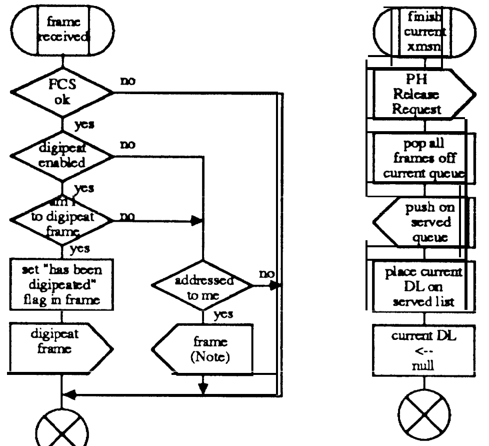
Link Multiplexor

Seized State -- State 2

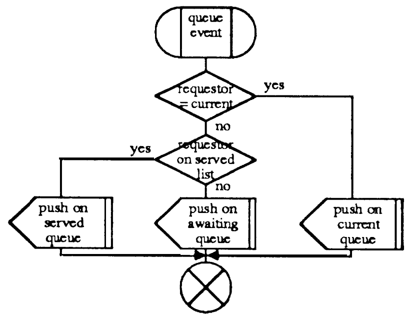


Link Multiplexor

Subroutines



Note -- The frame primitive is sent to the data link machine which is responsible for communications with the indicated remote (source) station.



Simplex Physical Layer State Machine

Eric L. Scace K3NA
10701 Five Forks Road
Frederick MD 21701
USA

0. Summary

This paper is part of a series of papers which provide extended finite state machine representations for AX.25 and related protocols. The state machines are depicted using state description language (SDL) graphic conventions from the 2.100 series of Recommendations developed by the International Telegraph and Telephone Consultative Committee (CCITT) of the International Telecommunications Union (ITU). An extended finite state machine representation of a communications protocol such as AX.25 avoids the ambiguities associated with prose descriptions. These descriptions also compel1 the protocol designer to confront many of the error scenarios which arise on a communications path, and simplify the implementor's task of producing correct solutions which will interwork with solutions created by others.

This particular paper describes an extended finite state machine suitable for use on shared simplex radio channels; such channels represent the vast majority of environments where AX.25 is employed today. A sister paper describes a similar machine suitable for use on full duplex channels.

1. Status of Proposal

The half-duplex physical SDL description here is a draft. The ARRL Digital Committee intends to include this machine as an Annex of the upcoming publication of AX.25 Revision 2.1.

Written comments have been received by the Committee which note the lack of material in the present description of AX.25 (i.e., Revision 2.0) which describe important parameters and operating procedures associated with half-duplex radio channels. In the dim dark days of the past, the original intent of the Committee was to focus the Rev. 2.0 publication strictly on the link layer protocol aspects of AX.25. It now seems prudent to include additional helpful information, not related to the link layer protocol in the strict architectural sense, which would be helpful in the correct understanding and implementation of AX.25 over radio channels.

The following material is still in a draft state. You are invited to review and comment on this material. Comments are desired so that the final publication is as useful as possible to its readers.

2. Features of the Simplex Physical SDL Machine

The simplex physical SDL machine includes the following features:

a) preferential treatment for frames to be digipeated, including a DWAIT timer.
b) guard times for changeover between receive and transmit.
c) guard times for a traditional repeater (i.e., AXDELAY and AXHANG).
d) p-persistence algorithm to improve channel utilization under heavy load (suggested by Chepponis & Karn at the 6th Computer Networking Conference).
e) 10 minute transmitter protection limit.
f) an independent anti-hogging function which relinquishes control of the channel after a user-specified period of time.
g) independent queuing of frames being digipeated and "normal" frames to be transmitted.

3. Location in Overall Model

This SDL machine resides at the physical (lowest) Layer of the standard Open Systems Interconnection reference model. It interacts with the link multiplexor SDL machine above it, and directly with a typical radio transceiver below it.

3.1 Interaction with the Link Multiplexor

The Link Multiplexor SDL machine directs the operation of the simplex physical SDL machine through the primitives described below. "PH" in some primitive names stands for "physical".

PH Seize Request -- This primitive requests the simplex physical SDL machine to begin transmitting at the next available opportunity. When that opportunity has been identified (according to the CSMA-CD/p-persistence algorithm included within), the transmitter started, a parameterized window provided for the start-up of a conventional repeater (if required), and a parametrized time allowed for the synchronization of the remote station's receiver (known as TXDELAY is most implementations), then a **PH Seize Confirm** primitive is returned to the Link Multiplexor.

Normal Frame -- This primitive from the Link Multiplexor SDL machine provides an AX.25 frame of any type (UI, SABM, I, etc.) which is to be transmitted. An unlimited number of frames may be provided. If the transmission exceeds the lo-minute limit or the anti-hogging time limit, the half-duplex physical SDL machine automatically relinquishes the channel for use by other stations. The transmission is automatically resumed at the next transmission opportunity indicated by the CSMA-CD/p-persistence contention algorithm.

PH Release Request -- The Link Multiplexor SDL machine provides this primitive when the submission of a sequence of frames to be transmitted on behalf of a particular AX.25 connection has been completed. The simplex physical SDL machine will then piggyback any straggling digipeat frames (if time permits) and then relinquish the channel.

Digipeat Frame -- This primitive from the Link Multiplexor SDL machine provides an AX.25 frame which is being digipeateci. The simplex physical SDL machine gives preference to digipeated frames over normal frames, and will take advantage of the DWAIT window. Digipeat frames can be provided by the Link Multiplexor at any time; a PH Seize Request and subsequent PH Release Request are not employed for digipeating.

During reception, the simplex physical SDL machine provides each AX.25 frame to the Link Multiplexor in a **Frame** primitive. No analysis is done on the frame by the simplex physical SDL machine; it does not. examine lengths, the frame check sequence, the need for digipeating, or any other content of the frame; these responsibilities are carried out by the higher level SDL machines.

In addition, the simplex physical SDL machine provides **PH Busy Indication** whenever the channel becomes busy. "Busy" here means the detection of a valid modem synchronization sequence, HDLC flags, or HDLC frames; it does *not* mean FM carrier detection on a 2-meter radio! The assumption here is that the channel (if FM) is not shared by voice users (although such sharing would be possible by expanding the meaning of "busy" in this sense) An indication of busy is provided to the higher layer SDL machines so that various timers which supervise the AX.25 connection can be suspended. This avoids the undesirable situation on a busy channel where AX.25, having sent data and expecting an acknowledgement, times out and attempts retransmissions -- and the only reason an acknowledgement was not received was because the remote station did not yet have a chance to make a transmission. **PH Quiet Indication** is provided when the channel becomes quiet again.

Since the channel is simplex, the PH Busy Indication and PH Quiet Indications are also provided when the simplex physical SDL machine causes a transmission to occur.

3.2 Interface to the Radio

As the lowest layer (in an Open Sys terns Interconnection achitectural sense) machine within a TNC, this machine is envisioned to manipulate a typical radio transceiver.

Turn On Transmitter and **Turn Off Transmitter** primitives are used to manipulate the transceiver's PIT line.

The simplex physical SDL machine sends a **Frame** primitive represents the actual transmission of a frame. Although SDL representation of bit-by-bit transmission of the contents of the frame are possible, they are not used here because the additional complexity was not required. The Frame primitive, however, differs from all other primitives used in these SDL machines in one respect: it is not atomic. Under this model, the Frame primitive occupies time; this allows the simplex physical SDL machine to consume time associated with transmission, and to trigger the 10 minute transmitter protection and anti-hogging timers.

Similarly, the simplex phsyical SDL machine employs a simple model of reception which collapses bit-by-bit reception of AX.25 frames into a single incoming primitive called **Frame.** The detection of modem synchronization, flag fill, or frame structure trigger the **Acquisition of Signal** primitive. The loss of modem synchronization, flag fill, or framing triggers the **Loss of Signal** primitive.

4. Internal Operation of the Machine

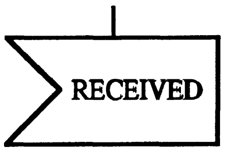
The internal states, queues, flags, and timers are summarized on the first page of the SDL diagram. All queues are first-in first-out queues. These items are used in a straightforward manner, so no further explanation will be provided here.

It should be noted that the anti-hogging time limit is not applied to the digipeating function. However, the lo-minute transmitter timer is enforced while digipeating. In the unlikely event that the lo-minute limit is exceeded, the transmission of digipeated frames is temporarily suspended and the channel relinquished. After other stations have had the opportunity to digipeat frames (i.e., DWAIT expires), but before the p-persistence algorithm kicks in, the SDL machine jumps back on the channel to resume transmission of those frames still in the digipeat queue. While this logic: is provided in the SDL diagrams for completeness, it seems unlikely that it would ever be utilized.

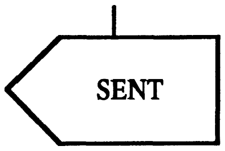
SIMPLEX PHYSICAL

Summary of Primitives, States, Queues, Flags, Errors, and Timers

PH Primitives

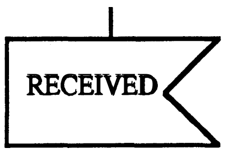


PH Sieze Request
PH Release Request
Digipeat Frame
Normal Frame



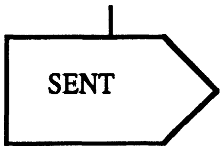
PH Seize Confirm
PH Busy Indication
PH Quiet Indication
Frame

PH Primitives



Acquisition of Signal
Loss of Signal
frame

Note -- Acquisition and loss of signal do NOT refer to FM carrier detect, but rather to modem synchronization, HDLC flags, and HDLC framing.



Turn On Transmitter
Turn Off Transmitter
Frame

Note -- Frame primitive is not atomic; it consumes the time needed to actually transmit the frame.

States

0 -- Ready
1 -- Receiving
2 -- Transmitter Suppression
3 -- Transmitter Start
4 -- Transmitting
5 -- Digipeating
6 -- Receiver Start

Error Codes

No error codes used.

Queues

Digipeat Queue -- holds all frames to be digipeated in the order in which they arrived from the higher layer.
Normal Queue -- holds all normal frames, plus Seize and Release Requests, in the order in which they arrived from the higher layer.

Flags & Parameters

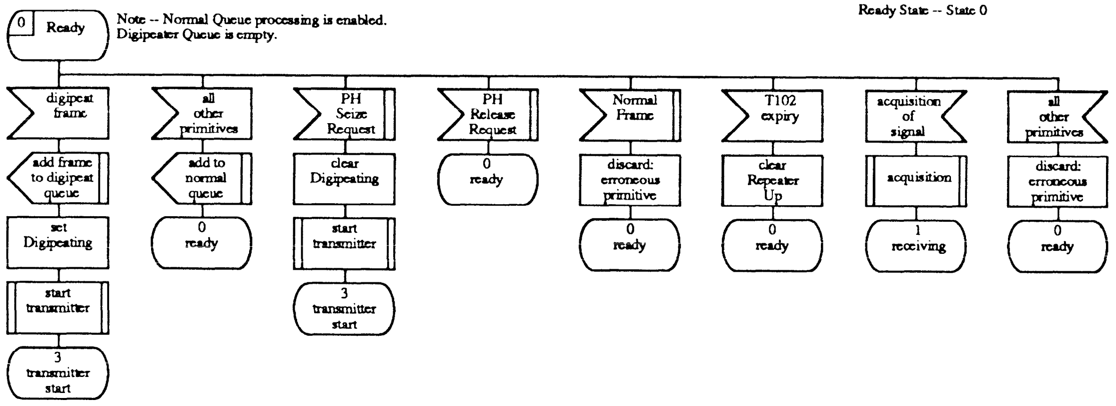
Digipeating -- Set when this transmission is for digipeating frames. Cleared when this transmission is for normal frames.
Repeater Up -- Set when repeater is expected to still be transmitting. Cleared when repeater carrier is expected to have dropped.
Interrupted -- Set when anti-hogging or 10 minute transmitter limits have interrupted the transmission of normal frames.
p -- p-persistence value, in the range 0-1.

Timers

T100 -- repeater hang (AXHANG).
T101 -- digipeater window (DWAIT)
T102 -- slot time (p-persistence)
T103 -- transmitter startup
T104 -- repeater startup (AXDELAY)
T105 -- remote receiver sync (TXDELAY)
T106 -- 10 minute transmission limit
T107 -- anti-hogging limit
T108 -- receiver startup

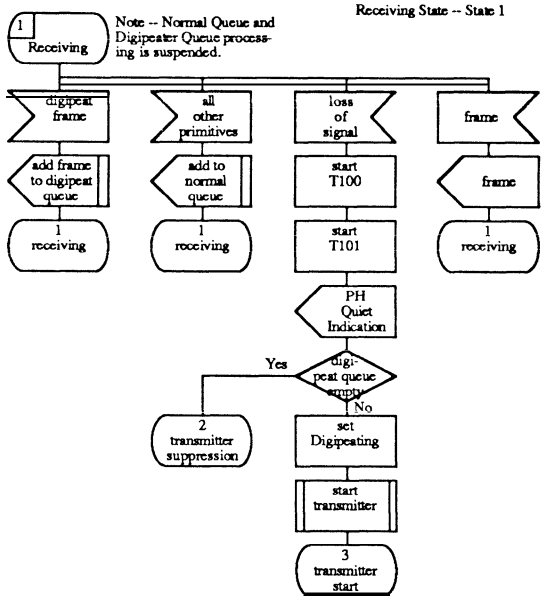
Simplex Physical

Ready State -- State 0



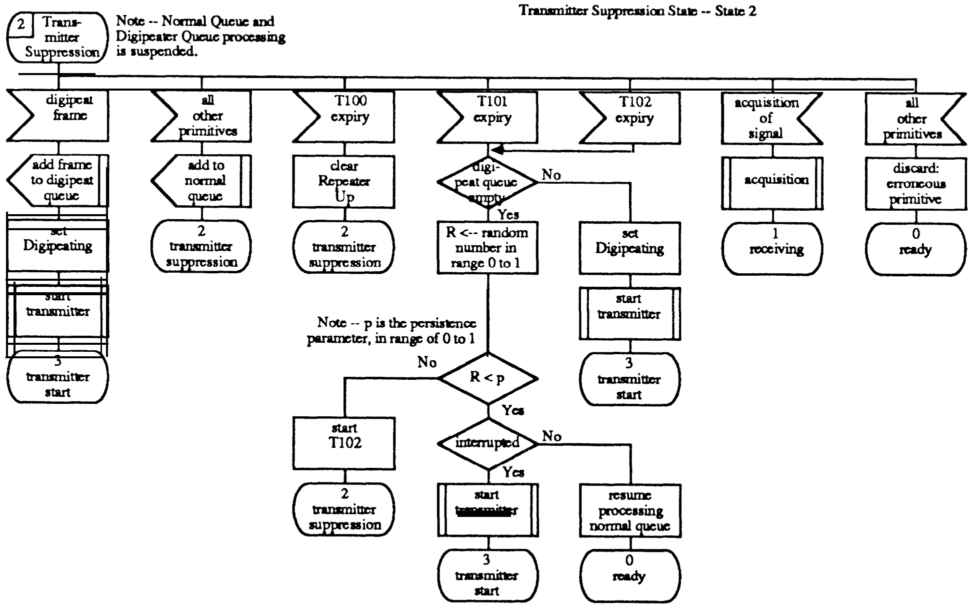
Simplex Physical

Receiving State -- State 1



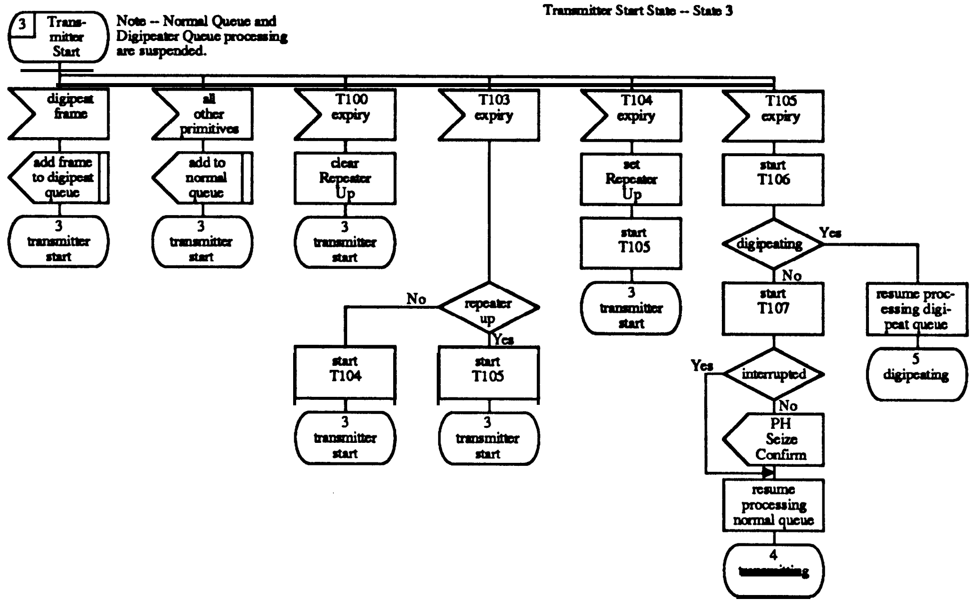
Simplex Physical

Transmitter Suppression State -- State 2



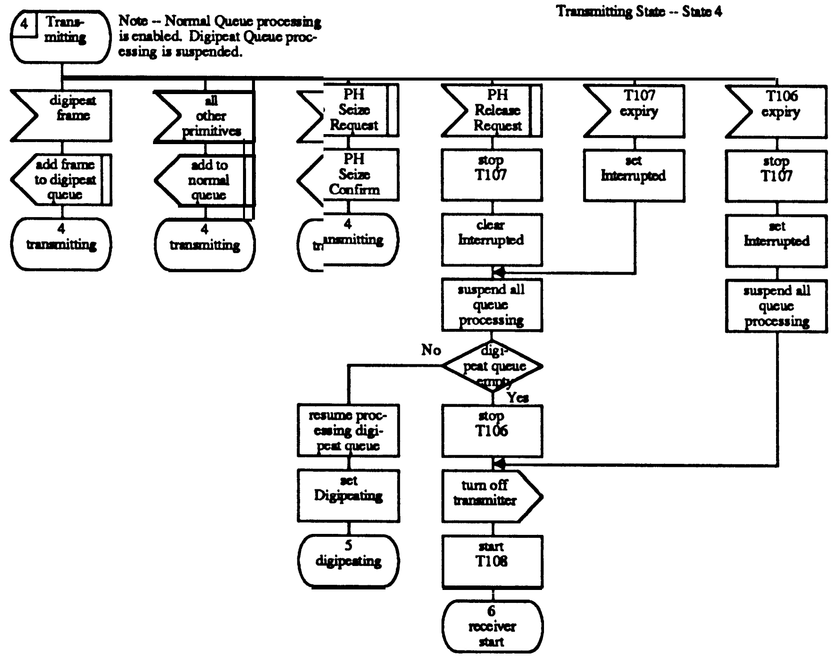
Simplex Physical

Transmitter Start State -- State 3



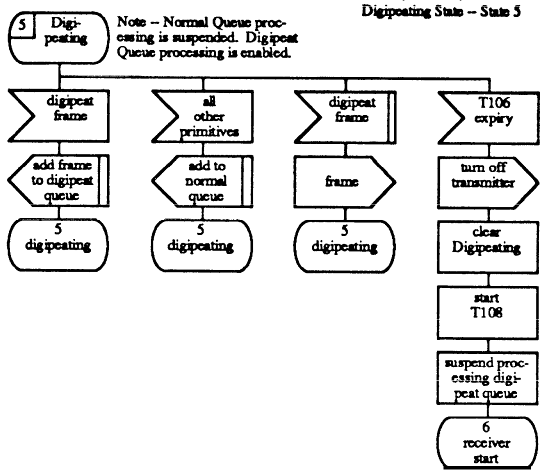
Simplex Physical

Transmitting State -- State 4



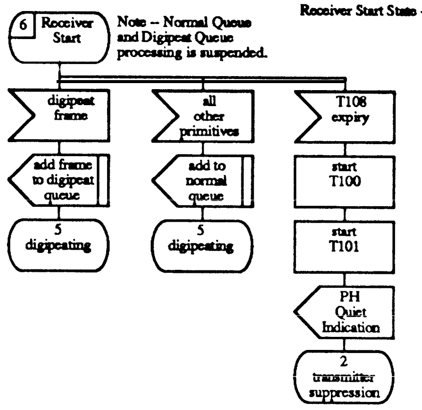
Simplex Physical

Digipeating State – State 5



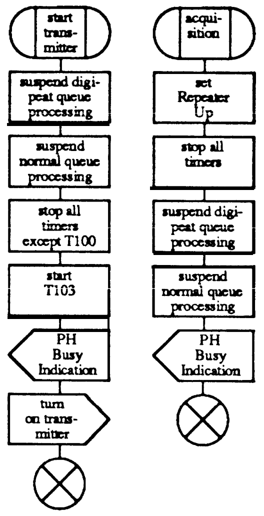
Simplex Physical

Receiver Start State – State 6



Simplex Physical

Subroutines


start trans-mitter
suspend digipeat queue processing
suspend normal queue processing
stop all timers except T100
start T103
PH Busy Indication
turn on trans-mitter
acqui-sition
set Repeater Up
stop all timers
suspend digipeat queue processing
suspend normal queue processing
PH Busy Indication

A BRIEF NOTE PROPOSING NON-ALOHA ACCESS TECHNIQUES FOR PACSATS

Jeff W. Ward, G0/K8KA
Research Fellow
UoSAT Spacecraft Engineering Unit
University of Surrey
Guildford, UK

ABSTRACT

Carrier-sense multiple-access (CSMA), as used in most terrestrial packet radio networks, is not efficient for low-Earth orbiting store-and-forward packet satellites (PACSATs). This note describes a simple time-division multiple-access protocol for PACSATs. A procedure is proposed for early experiments on the UoSAT-D Packet Communications Experiment (PCE) transponder, using AX.25 as the link protocol with satellite-controlled TDMA arbitration.

1. ALOHA UPLINKS

A low-Earth orbiting PACSAT will appear to users as a PBBS on a very high hill, and it will have the same access problems as a hilltop digipeater or PBBS: the PACSAT will hear many stations, but the user stations will not hear one another. The groundstations are all hidden terminals, and CSMA will not stop them from transmitting simultaneously on the satellite uplink. The uplink will tend to look like an ALOHA channel, modified by the FM capture effect. (Capture effect will increase throughput, as stronger stations override weaker ones during collisions on the uplink.) In the classical analysis, where all packets in a collision are destroyed, maximum throughput of an ALOHA channel is 18%, and this begins to drop toward nil when the channel is overloaded. See Tanenbaum (Ref. 1) for the bad news.

A simple solution to this problem - proposed for amateur PACSATs and used successfully on FO-12 - is to have several uplink channels and a single downlink. The ratio of 4 uplinks to a single downlink brings our theoretical maximum uplink throughput to 72% of the data rate. (This leaves some downlink bandwidth free for telemetry broadcasts, acknowledgements and multi-destination messages.) This simple solution works, but at a price. The satellite must have four data demodulators and four receiver IF chains, and the PACSAT computer must have the power to handle 4 uplinks simultaneously. Over lightly-populated areas, there may be only one station per uplink, and the combined uplink data rate will exceed the the theoretical 'maximum' throughput by a factor of five. Although this is not a problem for the new generation of PACSAT CPUs, it will be when data rates rise significantly above 9.6 kbps. From feast over the boondocks, we get famine in the cities; when the satellite is over a heavily populated area, the independent ALOHA channels will still tend to get overloaded, and throughput will fall. This is especially true when the satellite first comes into range of a population center and everyone starts trying to send messages at once.

Even if we accept these performance problems and use four uplinks per satellite, frequency allocation and band crowding will drive us to find more efficient access techniques. Relatively low Doppler shift make the VHF and UHF bands the natural home for low-Earth orbiting satellites, and soon we won't be able afford four VHF uplinks for every PACSAT. FO-12 and Microsats A & B already occupy twelve 2-meter user access channels. Hoping to ease this crowding problem, the Packet Communications Experiment (PCE) on UoSAT-D will be used to experiment with non-ALOHA techniques on a single uplink.

2. OTHER ACCESS TECHNIQUES

There are many multiple access techniques other than ALOHA, some of which are addressed in the references. When considering these schemes, keep in mind that the PACSAT environment is a new one for Amateur packet networking, because the satellite itself is a reliable, powerful central node which can form the hub of a non-ALOHA network. This central processing power should manage the available bandwidth efficiently and fairly. We should make sure that some traffic gets through even when the 'offered load' is high, and that small, weak stations don't get out-gunned by stronger ones.

Busy tone multiple access (BTMA) was considered. Whenever the uplink is bwsy, the satellite would transmit a 'busy signal' on the downlink telling other stations not to begin transmitting. We could not find a way to include a reliable, up-todate busy signal on the downlink without great complication and/or increased downlink bandwidth. We also ruled out spread-spectrum techniques such as code-division multiplexing, although these may play a role in future amateur packet satellites.

We decided to concentrate on time-division multiple access (TDMA) for the time being. The UoSAT-D PCE will act as master station and groundstation PCs or TNCs will be the slaves. 'TDMA' does not imply that groundstations will have to be locked to a common time base, transmitting their data in precisely synchronized bursts. The term indicates that the PCE will manage groundstation access by dividing the uplink into time slots, with different slots for different uses. We will attempt to adapt the TDMA system to AX.25 link layer so as not to completely remove the installed "user base" who have AX.25 TNCs. (At the very least, KISS TNCs could be used as HDLC frame generators.)

Harold Price first drew my attention to such a scheme in a study report about the Swedish MAILSTAR satellite, and subsequent conversations with Phil Karn developed the ideas and the analogy to HF DX operations.

3. THE PCE ACCESS SYSTEM

All communications between groundstations and the UoSAT-D PCE will be computer-to-computer transfers. Message system operations like List, Read, Send and Delete will be high-speed, full-duplex transactions between the PCE and a groundstation computer. The human interface -which presents message lists clearly, compresses and expands text messages, and provides the user with a nice menu - will be in groundstation software. While this shifts the user interface from one satellite into thousands of user computers, we believe

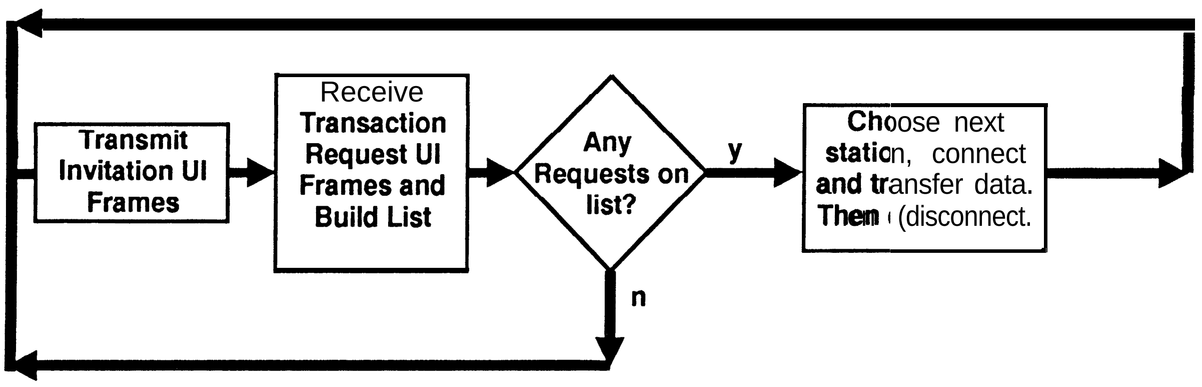

Transmit Invitation UI Frames
Receive Transaction Request UI Frames and Build List
Any Requests on list?
y
n
Choose next station, connect and transfer data. Then disconnect.

Fig. 1 - Flow Chart of TDMA with ALOHA Request Period

that the groundstation computer has more time and memory for these tasks than the satellite computer does.

The protocol described below is shown as a flow chart in Figure 1.

3.1 Pile Up Operation

Whenever the PCE is idle, it will transmit an **Invitation** frame which invites groundstations to reply. This will be an AX.25 UI frame identified by its contents and/or a special PID. Upon hearing this, the groundstation computer will start a random **backoff** timer, after which it will transmit a **Transaction Request** packet on the **uplink.** When the PCE hears one of these **Transaction Requests,** it connects to the station heard and begins a message transaction using a standard **AX.25** link. When the transaction is complete, the PCE goes back into the idle state.

This protocol divides **uplink** time into two portions: an ALOHA portion during which all groundstations can transmit, and a contention-free portion during which one groundstation has the entire **uplink** to itself. The theoretical maximum throughput of this system and is:

1 - (ALOHA time **/** total time).

3.2 List Operation

The PCE will probably hear Transaction Request frames from several stations during the ALOHA window. It will keep a list of stations heard and then work them one at a time in successive contention-free transactions. The **Transaction Request** frame transmitted by a groundstation will **carry** information to help the PCE selection algorithm choose a station for the next transaction. A few factors which might be included in a priority scheme are:

- the length of the requested transaction,
- message priority,
- groundstation priority (higher for command stations or emergency stations),
- time until groundstation loss of satellite,
- time until next acquisition of satellite at this groundstation, and
- time since last transaction with this groundstation.

Obviously, the choice of priority factors and the calculations made by the PCE determine how the available pass time will be shared amongst many groundstations trying to use the PCE. Now we can start tweaking. Priority factors can be optimized to give greater throughput, greater equality of access, more efficient use of groundstation transmitters, etc. They could even be dynamically modified by an "expert system" in the PCE. This certainly beats the complete lack of "**tweakability**" available in the ALOHA system.

3.3 Token Passing

The TDMA technique described here can also be viewed as a token passing protocol. The satellite always passes the token to a groundstation, and the token will revert to the satellite when a transaction is completed or a connection fails. In a fullduplex environment, connection timeouts can be quick, increasing efficiency of the token passing. Efficiency will also depend on variables such as groundstation transmitter keying times, average message length, the priority algorithms and the RF link quality.

4.0 CONCLUSION

A successful **TDMA/Token-passing** protocol for PACSAT access will promote efficient use of **uplink** channel RF bandwidth and make best use of satellite on-board computing power. The transaction-based scheme proposed here could have several advantages over ALOHA:

- Weak stations and strong stations compete for throughput on an equal basis.

- Protocol variables can be adjusted to optimize a chosen parameter.

- Even when the offered load is much greater than can be handled by the uplink, some messages will get through.

The UoSAT-D Packet Communications Experiment will provide a platform for protocol experiments, not simply a communications service. The multitasking operating system (Quadron Communications Facility - qCF) which is being developed for UoSAT and Microsat will be an ideal environment for these experiments. Its high-level language support will speed development and debugging of protocol software, and a multi-tasking system keeps experiments away from spacecraft housekeeping tasks which will also be running on the PCE.

At UoS we are now finalizing PCE hardware design, while Harold Price and Skip Hansen are modifying qCF for spacecraft use. Applications software development will begin with spacecraft housekeeping, a mailbox, and the simple protocol outlined above. Groundstation software will be developed simultaneously, and we would like to have the entire system operational by launch. If the launch is in January, as currently scheduled, this will be tough (read “impossible”), but if the launch is delayed until June, there is hope.

REFERENCES

1 . Tanenbaum, AS., 1981, “Computer Networks”, Prentice Hall, Englewood Cliffs, NJ.

2. Karn, P., 1987, "A High Performance, Collision-Free Packet Radio Network”, ARRL 6th Computer Networking Conference Proceedings, Redondo Beach, USA.

3. Fox, T., 1986, "RF, RF, Where is My High Speed RF?“, ARRL 5th Computer Networking Conference Proceedings, Orlando, USA.

THE UOSAT-D PACKET COMMUNICATIONS EXPERIMENT

Jeff Ward, G0/K8KA
Research Fellow
UoSAT Spacecraft Engineering Research Unit
University of Surrey
Guildford, United Kingdom

ABSTRACT

This paper describes the Packet Communications Experiment (PCE), which will be the primary payload on the UoSAT-D satellite. UoSAT-D is to be launched in 1989, along with several other Amateur PACSATs. The PCE design will be based on results of the UoSAT-OSCAR-11 Digital Communications Experiment, using an 80C186, 512 kbytes program RAM and 4 Mbytes message-storage RAM. It will be an open-access Amateur Radio PACSAT messaging system.

INTRODUCTION

Before the end of 1989, Amateur Radio will be able to boast five new packet radio satellites. The launch of an Ariane 40 rocket with the French SPOT-2 imaging satellite as primary payload will mark an unprecedented expansion in the number of Amateur Radio satellites in orbit. Six secondary payloads will be launched beneath SPOT-2: 4 Microsats built by AMSAT-NA and its collaborators, and 2 UoSATs built at the University of Surrey. At least five of of these satellites will have AX.25 packet radio data links, and several will be dedicated 'PACSATS' for store-and-forward packet radio message handling.

One of the new PACSATs will be UoSAT-D, carrying a Packet Communications Experiment (PCE) as its primary payload. The PCE is based on an 80C186 processor with 512 kbytes of program RAM and a 4-Mbyte RAMDISK for message storage. 9600 bits/sec FSK packet links will make this the fastest of the new PACSATs. The PCE will implement the AX.25 link protocol in support of a range of higher-level protocol experiments. The UoSAT-D PCE, described in this paper, will be an open-access PACSAT transponder for Amateur Radio stations.

1.0 RESULTS FROM THE UOSAT-OSCAR-11 DCE

Previous experiences with store-and-forward satellite operation have influenced the PCE design. In particular, the UoSAT-OSCAR-11 Digital Communications Experiment or DCE has provided valuable data (Ref. 1, 2, 3). The DCE, built by AMSAT volunteers in the USA and Canada, was included on UO- 11 to provide in-orbit tests of radiation effects on VLSI components and to prove that store-and-forward communications using low-cost satellites and small earth terminals was possible. The DCE has been a great success, providing unique engineering data, forwarding amateur packet messages internationally, and saving UoSAT-OSCAR-11 from a failed on-board data link.

The DCE hardware is an NSC-800 microprocessor, an 82C55 parallel I/O chip, and memories ranging in density from 4k X 1 bit to 8k X 8 bit. With exceptions detailed in Ref. 3, these devices have survived the radiation dose of 4 years in a 700 km polar orbit.

1 .1 Radiation-Induced Single Event Upsets

One of the most interesting aspects of the DCE mission has been the monitoring of cosmic particle induced bit-flips or single event upsets (SEUs) in CMOS static memory. SEUs in the memory protected by hardware error detection and correction (EDAC) and in the unprotected memory are monitored by DCE software.

In 144 kbits of EDAC-protected memory - composed of 4k X l-bit devices - 60 SEUs have been recorded in the last two years. The mean error rate estimate is:

5.98×10^{-7} errors/bit/day. [See Note 1]

In practical terms, this error rate means that you can expect your software to crash once per day if you have 240 kbytes of this memory with no EDAC hardware. The DCE's EDAC hardware has protected it from any such crashes.

SEU monitoring on the 8-bit wide memories began in earnest during 1988. This memory is allocated in 256-byte blocks for message storage. Each block is divided into two 128-byte 'words', each of which is protected by a 3-byte EDAC code. This code is two 8-bit CRCs and an exclusive OR sum. The CRCs (generated by table lookup for speed) locate errors, while the checksum gives the error pattern. This EDAC code can locate and correct any byte-wide error in a 128- byte word. Memory wash software periodically checks each word for errors, logging and correcting any which are found. Twenty one errors were found over the 48 days that this software has been fully operational. This implies an error rate of:

5.70×10^{-7} errors/bit/day.

This is strikingly similar to the rate for the bit-wide memories, indicating that the bit-wides and the denser byte-wides have approximately the same SEU sensitivity - contrary to what we might intuitively expect. SEU susceptibility is related to feature size and device geometry, and UoSAT is undertaking a collaborative research project with the European Space Agency (ESA-ESTEC) to use the UoSAT-2 SEU measurements in refining ground-based SEU models and component tests.

All of the errors observed in byte-wide memories have been single bit errors, and this significant observation has influenced the UoSAT-D PCE design. Previously, we thought that the dense packing of bits in byte-wide devices would mean that one cosmic particle might upset several bits in a single byte. Since multi-bit errors are not corrected by the hardware EDAC circuits (based on the Hamming (8,12) code), byte-wide memories were never used for spacecraft EDAC memory. The in-orbit observations of the DCE show that multi-bit errors are not common in byte-wide devices, implying that byte-wide devices can be used in EDAC memory for storing programs and critical data.

1.2 Messaae Forwardinq Experiments

While acting as a hardware test-bed, the DCE has also been used as a communications channel, linking Amateur Radio packet networks in the U.K., Australia, New Zealand and South Africa through BBS mail forwarding. A total of 10 gateway groundstations (including one in Antarctica and two in Pakistan) have been activated during the experiment, and 6 are now passing BBS traffic via UoSAT-2. Although this is a small population, it has provide some insight into the operation of satellite gateways for digital communication.

Three important observations follow:

- The introduction of reliable packet satellites which can be interconnected with the existing BBS networks will make Amateur Radio packet networks truly international. The current mail addressing and routing system breaks down when several international transport systems are available. We must quickly look for solutions to international addressing and routing problems.

- The DCE is data transparent, and it has frequently been used to forward binary files and archived text. Text archiving (using the common PKARC/PKXARC combination) provides compression of up to 60%. This simple procedure can have the combined effect of doubling a link's data throughput **and** doubling the memory available on the satellite. The packet radio community should adopt a suitable standard for compressed text and use it whenever possible.

oProtocols which are efficient for terrestrial BBS use are not necessarily best for communications with quickly-moving, low-Earth orbiting (LEO) satellites. The DCE MSG2 protocol (developed by Price and Ward) has proven quite efficient for moving groups of BBS messages in a single ‘transaction’ with the satellite. An important feature of this protocol is the ability to continue a ‘transaction’ on a later pass from the point at which you lose the satellite. UoSAT-D will be used to experiment with speciaiised LEO satellite protocols above the AX.25 link layer.

2 THE UoSAT-D PACKET COMMUNICATIONS EXPERIMENT

The UoSAT Unit intended to fly a packet communications system on the UoSAT-C satellite (Ref. 4), which was scheduled for launch by NASA/DELTA in 1988. Unfortunately, this launch and the UoSAT-C mission have been postponed. The SPOT-2 opportunity, jointly developed by UoSAT, AMSAT-NA and Arianespace, became the best time to launch the PCE. The PCE will be the primary payload on UoSAT-D, one of the two UoSAT satellites on the SPOT-2 launch.

UoSAT-D will be slightly smaller than other UoSAT-1, 2 & C satellites, roughly 345 mm square in cross section and 550 mm tall. The modular satellite structure is composed of stacked 345 X 345 X 28 mm ‘module boxes’, and the PCE occupies two of these boxes.

The design of an advanced store-and-fotward transponder at UoS is being funded by a contract from the Volunteers In Technical Assistance (VITA). VITA wish someday to use store-and-forward communications on non-Amateur frequencies to link their headquarters in Washington, DC with technical volunteers in remote rural areas of developing countries. The PCE hardware flown on UoSAT-D will be a prototype for the VITA payloads. Operational experience from the UoSAT-D Amateur Radio communications mission will be used to optimise the PCE design, and final operational payloads should be ready for a dedicated VITA satellite which could be launched within the next few years.

AMSAT-UK Is funding a substantial portion UoSAT-D mission.

2.1 UoSAT-D On Board Data Handlinq

Like UoSAT-1 and UoSAT-2, UoSAT-D will carry several on-board computers (OBCs) linked in an on-board data handling (OBDH) network. The PCE will act as the primary OBC, backed up by an 1802 similar to those used on UoSAT-1 and UoSAT- 2. The third OBC is an 80C30 microcontroller dedicated to a radiation monitoring experiment (Cosmic Particle Experiment / Total Dose Experiment - CPE/TDE). These OBCs will be linked by a multiple access serial bus using 9600 bit/sec asynchronous packets. Most of the data on the OBDH bus will be CPE/TDE output being sent to the PCE for storage and later packet downlinking.

Telemetry and teiecommand can function as stand-alone hardware systems or as peripherals to the OBCs. Telemetry is accessed through parallel links; the OBC generates a channel address and the telemetry system returns a 12-bit A-to-D converter value. The OBCs have serial inputs to the teiecommand system; 1200 bit/sec command packets with error detection information are sent to the command decoder, where they are verified, decoded and latched. This system combines the flexibility of computer driven telemetry and telecommand with the reliability of stand-alone hardware. This combination has proven itself on the previous UoSATs, where hardware probiems have been circumvented through alternative communications paths, and the missions are frequently enhanced by software upgrades.

3. PCE HARDWARE

The PCE is separated into two module boxes. One box contains the CPU, program memory and peripherals, while the second hdds a 4-Mbytes RAMDISK for message storage.

3.1 PCE CPU and Memory

The Packet Communications Experiment CPU module (PCE CPU) is an onboard computer intended to be the primary operational OBC on UoSAT-D. Parallel interfaces for telemetry, magnetometer, battery charge regulator A-to-D converter and RAMDISK are provided. Four serial channels serve telecommand, DASH and packet radio channels. (Fig. 1)

The PCE CPU is an iNTEL 80C186 microprocessor running at 7.3728 MHz. This processor, of unknown radiation tolerance, was chosen because it represents the state of the art in highly-integrated CMOS microprocessors (Ref. 5). It provides clock generator, Interrupt controller, DMA controller, timers, chip- select generation and a 16-bit 8086-compatible CPU in a single package. None of the processors for which we have radiation tolerance data provide this much function in such an efficient package.

The 80C186, with full 16-bit bus, was chosen to allow demanding multi-tasking experiments. As the primary OBC on UoSAT-D, the PCE will execute the following tasks:

- closed-loop attitude control, using magnetorquing algorithms which will keep the satellite Earth-pointing to within 5 degrees,
- power conversion optimisation,
- data collection and storage for the radiation experiments,
- long-term telemetry surveys,
- experiments in on-board expert systems and satellite autonomy,
- SEU detection and correction.

Of course, PCE will also provide 9600 bit/second packet communications. The high bus bandwidth of the 80C186 is justified in this demanding environment. The word-wide bus requires more power and PC board area than the 80x88-style byte-wide bus, but this was considered a reasonable tradeoff given the size of the UoSAT-D PC boards and the power available. The 80C186 is also a more mature product than the 80C188, and it is available as an extended temperature range device.

The PCE CPU board contains 256 kbytes of EDAC RAM for programs and critical data (e.g. the file allocation tables for the RAMDISK). This EDAC memory will use 32K X 8 chips, since the results of SEU monitoring on the UO-11 DCE indicate that byte-wide memories can be successfully protected by single-bit correcting codes such as the Hamming (12,8). Each 32K words of EDAC RAM will consist of 4 memory chips, two for the 16 bits of data and one each for the two 4-bit Hamming check codes. Although consideration has been given to storing the two check codes in one device, this is unduly complicated by the need for 80C186 to make byte-wide writes to memory. Separate error counters will signal errors detected in the high-byte and low-byte memory banks. The EDAC system can be disabled by spacecraft telecommand in case of failure in the detection or correction circuits.

In addition to the EDAC RAM, the PCE CPU will have 256 kbytes of unprotected RAM. Assuming that the SEU rate per bit is similar to that observed on UoSAT- OSCAR-l 1, this unprotected nnemory can be used safely for temporary data storage.

The RAMs on the PCE CPU board will be from three different manufacturers, in three different package types. 128 kbytes of EDAC RAM will be standard .6-inch wide DIPs, and the other 128 kbytes will be .3-inch 'skinny-DIPs'. The non- EDAC RAM will be two Hybrid Memory Systems hybrids, each made with four surface mounted 32k X 8 chips on a single carrier. This mixture provides reliability through dissimilar redundancy.

3.2 Input/Output

3.2.1 Serial Channels

The 85C30 serial communications controller (SCC) was selected for the PCE because it supports both synchronous and asynchronous serial communications, it has an integral baud-rate generator, and it is well understood by AMSAT programmers. The 80C30, although a faster and more easily programmed device, was not available as an extended temperature-range part at the time the PCE was designed. There are two SCCs providing a total of 4 serial channels for packet radio and OBDH.

The PCE requires a flexible serial system, in which either SCC can be used for any of the possible communications tasks. Reliability is inversely proportional to complexity, however, so a comprehensive array of digital selector chips Is not used. One channel on each SCC has a digital selector on its input, and the other channels are directly wired to data sources. This system is a compromise between complexity and flexibility.

The downiink HDLC channels are the most sensitive to service timing - too much interrupt latency on these channels will cause transmitted frames to be aborted. Thus, either of the 80C186 DMA channels can be used as an HDLC output channel. The other DMA channel can be switched between HDLC receiving or RAMDISK data transfer.

The telecommand interface is a 1200 bit/sec. asynchronous serial channel using 8 data bits and an parity bit. For reliability, the telecommand channel is directly connected to the data output of one of the SCCs.

The fourth SCC channel is connected to the shared serial OBDH network. This channel will normally handle data packet bursts at 9600 bits/sec. If any of the PCE data links fail, however, the OBDH network connection can be used as a link to almost all data-generating and receiving devices on the satellite - particularly to the receivers and transmitters.

3.2.2 Parallel Links

The parallel i/O system uses 3 Harris 82C55 PPIs. The 82C55 in the DCE is still functioning, giving us some confidence in this chip. The nine 8-bit parallel ports provided by these chips are used for the telemetry system, the RAMDISK interface, the navigation magnetometer, and the BCR D-to-A converter.

The DAC deserves further comment. it is a current-output DAC (AD7524) driving a variable set-point input on the battery charge regulator (BCR). The optimum operating point for the BCR is influenced by battery charge state, temperature and solar panel illumination. The PCE will measure this point every five minutes by varying the DAC value and using the telemetry system to watch the solar-panel power output. The DAC value will then be set at the optimum point until the next measurement is made.

The remaining parallel I/O bits are used for EDAC counters, digital selector control and downlink keying.

3.3 The RAMDISK

An operational store-and-forward transponder must provide several megabytes of memory for message storage. SYSOP chores will become onerous and reliability will suffer if too little memory is provided. There are several methods of providing expanded memory within the limited physical address space of the selected microprocessor. in the UoSAT-OSCAR-11 DCE, bank switching is used, with four banks of memory selected by a parallel port to appear at the top of the processor memory map. The 1802 OBC on UoSAT-OSCAR-11 uses the 192 kbytes of RAM in the Data Store and Readout (DSR) as a sequential memory through a 1200 bit/sec serial link. The UoSAT-D PCE will use a parallel link between the 80C186 CPU and a 4-Mbyte RAMDISK module. Although memory-mapped, bank switched mass storage was originally proposed for the PCE, this posed power consumption and redundancy problems. If the 4 Mbytes were in the address space of the processor, they would have to be organised as a 16-bit wide memory resulting in increased power consumption and PC board area. A dedicated memory would also be [difficult to share between two processors, ruling out the possible inclusion of a backup processor. So, a RAMDISK with parallel interface was chosen.

The RAMDISK is divided into individually addressable sectors of 256 bytes each, and the interface is similar to that used by a disk drive. To store data in a sector, the PCE CPU uses control lines to command the RAMDISK into write mode, sends two bytes of sector address, and then sends the data. To read a sector, the PCE CPU commands the RAMDISK into read mode, sends the sector address and then reads the desired data bytes. Contiguous sectors can be read or written without re-addressincl the RAMDISK. A bidirectional parallel bus with handshaking and control lines connects the RAMDISK and the PCE CPU. If a second microprocessor were available, the RAMDISK bus could pass through a digital selector controlled by spacecraft telecommand.

In a 4 Mbyte memory system, it is possible that cosmic particles will cause CMOS latchup, or that something (e.g. a failed by-pass capacitor) will result in undesirable power consumption in part of memory. To keep a single failure such as this from disabling the entire system, the memory is divided into four units. Each unit is provided with its own power supply, so that each can be powered down if it fails or is not needed. Each unit is also buffered from the common address bus. One Mbyte of memory will be 128-kbyte hybrids as used in the CPU RAM, the remaining memory will use 256-kbyte SIL hybrids. The control circuits for the RAMDISK are implemented in HCMOS, designed for full-speed (1 Mbyte/sec) DMA access by the PCE CPU.

4.0 RF DATA LINKS

UoSAT-D will use FSK uplinks and downlinks as proposed in Ref. 4, but using a Mode-J (70-cm downlink / 2-m uplink) bandplan. Mode-J was chosen after considerable experience with UoSAT-2 showed 70-cm to be significantly quieter at the groundstation than 2 meters. UoSAT-D is designed to provide good downlink data (2.5 X 10 ^-5 BER) to stations with modest antennas. This requires several dB groundstation antenna gain between 0 and 30 degrees elevation and decreasing gain as the satellite rises and range decreases. This pattern can be achieved by several simple, non-steered antennas. There will be a high-power downlink on the satellite for experiments with very small groundstations and a low power downlink for periods of poor power budget.

Data will be scrambled (to remove DC component) using the same polynomial as the K9NG/TAPR modem. This polynomial is also available on the G3RUH FSK modem.

A Mode-J frequency plan brings with it the problem of identifying uplink channels in the crowded 2-meter band. UoSAT-D and the Microsats will often be visible simultaneously, aggravating this frequency allocation problem. We expand the problem yet again if we use a simple cure for the inefficiency of ALOHA access: four uplinks for each satellite. UoSAT-D will use only one uplink, and it will not have ALOHA access. Possible access techniques are discussed in an accompanying paper “Alternatives to ALOHA for PACSAT Access.”

5.0 SOFTWARE

One of the reasons for deciding on the 80C186 CPU was the ready availability of software. The basic multitasking kernal is being supplied by Quadron Services Corporation. The non-mission-specific kernal services are being ported from Quadron’s existing multi-tasking system with desiqn input from UoSAT and AMSAT-NA. There is a paper in these proceedings by H Price and B McGwier describing the kernal and the Microsat applications software. UoSAT-D applications programs for packet communications and spacecraft housekeeping will be developed at UoSAT.

6.0 CONCLUSION

The simultaneous launch of 5 OSCARS with packet radio capability is an exciting and interesting prospect. **UoSAT-D** will be carrying a PACSAT transponder similar in function to those on the Microsats, yet this transponder will use different hardware, different link speeds and different access techniques. This is an ideal opportunity for experimentation, and it provides in-orbit redundancy never before experienced by **AMSAT.** Once proven in orbit, **80C186** PCE will be a high-performance OBC design for for low-Earth orbiting satellites. Through transfer of PCE technology to VITA, Amateur Radio will again have proven itself capable of conducting state-of-the-art research and design with widespread applications.

ACKNOWLEDGEMENTS

I am greatly indebted to the North American team who built the **UoSAT-OSCAR-11** DCE and to all of the spacecraft operators and programmers **UoSAT;** The DCE gateway operators have also put long hours and personal funds into the project. MS Hodgart designed the code used to protect the DCE **RAMUNIT from SEUs,** and Harold Price & Skip Hansen continue to provide expert advice on the design of PCE hardware and software. Substantial funding came from UK SERC, ESA-ESTEC and the University of Surrey.

NOTES

Note 1: SEU occurrence is assumed to have a Poisson distribution, and the 5% to 95% probability bounds for the measured mean error rates are:

Bit-wide memories **(HM-6564)**	4.99×10^{-7} to 7.48×10^{-7}
Byte-wide memories (Hitachi 6616 **&** 6264)	4.07×10^{-7} to 7.87×10^{-7}

REFERENCES

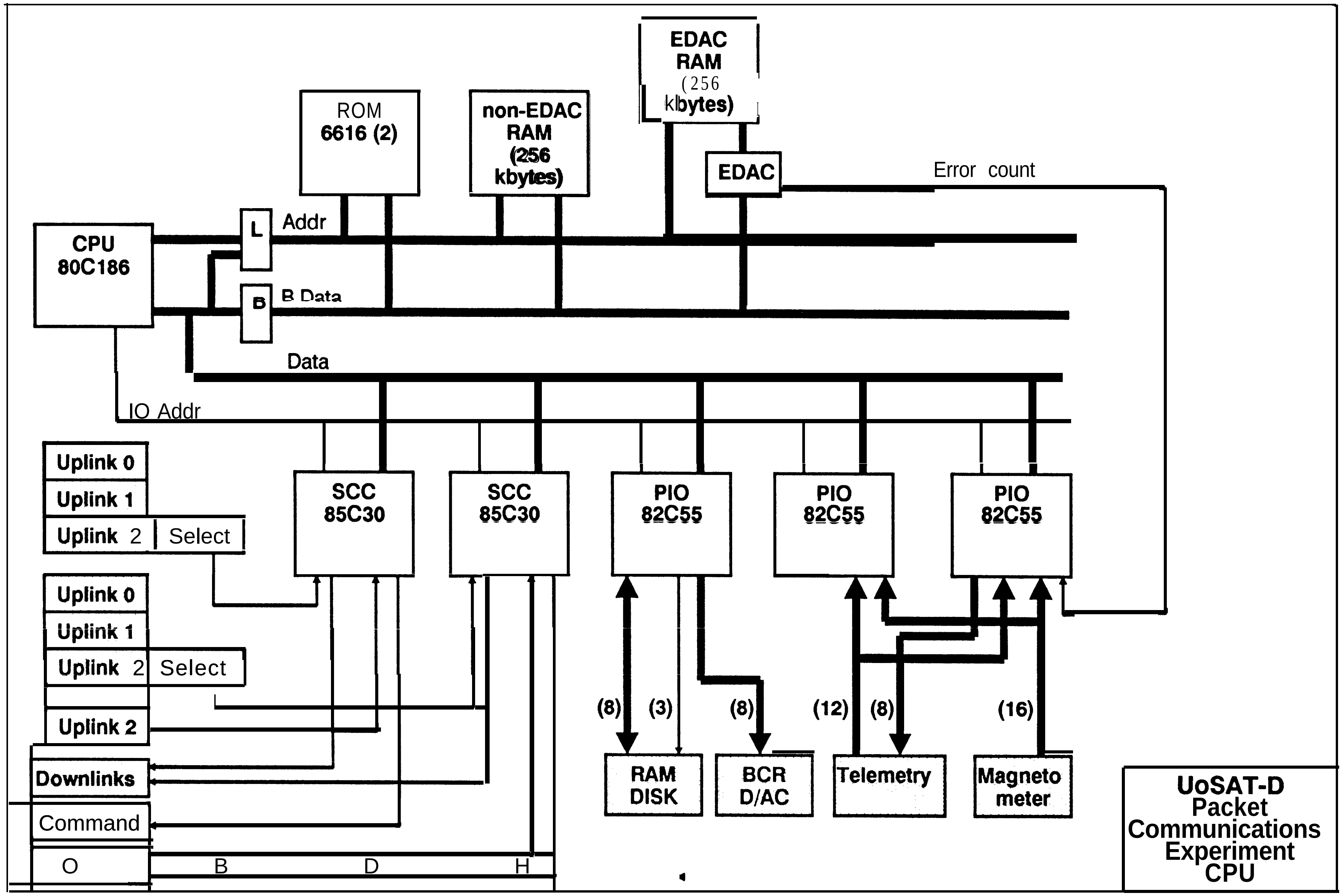
1. Johnson, L, 1984, ‘The OSCAR-l 1 Packet Experiment”, ARRL 3rd Computer Networking Conference Proceedings, Trenton, USA.

2. Ward, J. and Price, H., 1986, "The UO-11 DCE Message Store-and-Forward System”, ARRL 5th Computer Networking Conference Proceedings, Orlando, USA.

3. **Ward,** J. and Price, H., 1987, “The UoSAT-2 Digital Communications Experiment”, The Radio and Electronic Engineer, 57, September/October 1987 (Supplement).

4. **Hodgart,** S. and Ward, J., 1986, “FSK Methods for PACSAT Communication”, ARRL 5th Computer Networking Conference Proceedings, Orlando, USA.

5. **Brock,** M., 1987, “A High Performance Packet Switch”, ARRL 6th Computer Networking Conference Proceedings, Redondo Beach, CA, USA.


EDAC RAM (256 kbytes)
ROM 6616 (2)
non-EDAC RAM (256 kbytes)
EDAC
Error count
CPU 80C186
L
Addr
B
B Data
Data
IO Addr
Uplink 0
Uplink 1
Uplink 2
Select
Uplink 0
Uplink 1
Uplink 2
Select
Uplink 2
Downlinks
Command
O
B
D
H
SCC 85C30
SCC 85C30
PIO 82C55
PIO 82C55
PIO 82C55
(8)
(3)
(8)
(12)
(8)
(16)
RAM DISK
BCR D/AC
Telemetry
Magneto meter
UoSAT-D Packet Communications Experiment CPU

Supplement to 7th Computer Networking Conference Proceedings

Eric L. Scace K3NA
10701 Five Forks Road
Frederick MD 21701
USA

Attached are three additional documents for the 7th Computer Networking Conference:

- AX.25 Data Link State Machine -- SDL Revisions & Corrections -- 1988 Sep 30
- Parameter Negotiation between Consenting AX.25 Stations
- Segmenter State Machine

AX.25 Data Link State Machine

SDL Revisions & Editorial Corrections 1988 Sep 30

Eric L. Scace K3NA
10701 Five Forks Road
Frederick MD 21701
USA

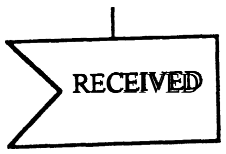
Attached are revisions and editorial corrections to the SDL diagrams which appeared in the 7th Networking Conference proceedings. You should find the following four changed pages:

- summary page
- Connected State, state 3 -- page 2 of 2
- Timer Recovery State, state 4 -- page 1 of 2
- Subroutines

DATA LINK

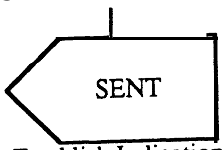
Summary of Primitives, States, Queues, Flags, Parameters, Errors, and Timers

DL Primitives



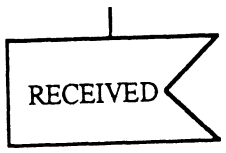
DL Establish Request

DL Release Request
DL Data Request
DL Unit Data Request
Set Own Receiver Busy
Clear Own Receiver Busy



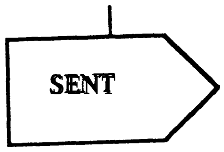
DL Establish Indication
DL Establish Confirm
DL Release Indication
DL Data Indication
DL Unit Data Indication
DL Error Indication

LM Primitives



LM Seize Confirm

DM
UA
SABM
DISC
RR
RNR
REJ
UI
FRMR
I



LM Seize Request
LM Release Request
DM
UA
SABM
DISC
RR
RNR
REJ
UI
I

States

0 -- disconnected.
1 -- awaiting connection.
2 -- awaiting release.
3 -- connected.
4 -- timer recovery

Queues

I frame queue -- queue of information to be transmitted in I-frames.

Timers

T1 -- outstanding I-frame or P-bit.
T3 -- idle supervision (keep alive).

Error Codes

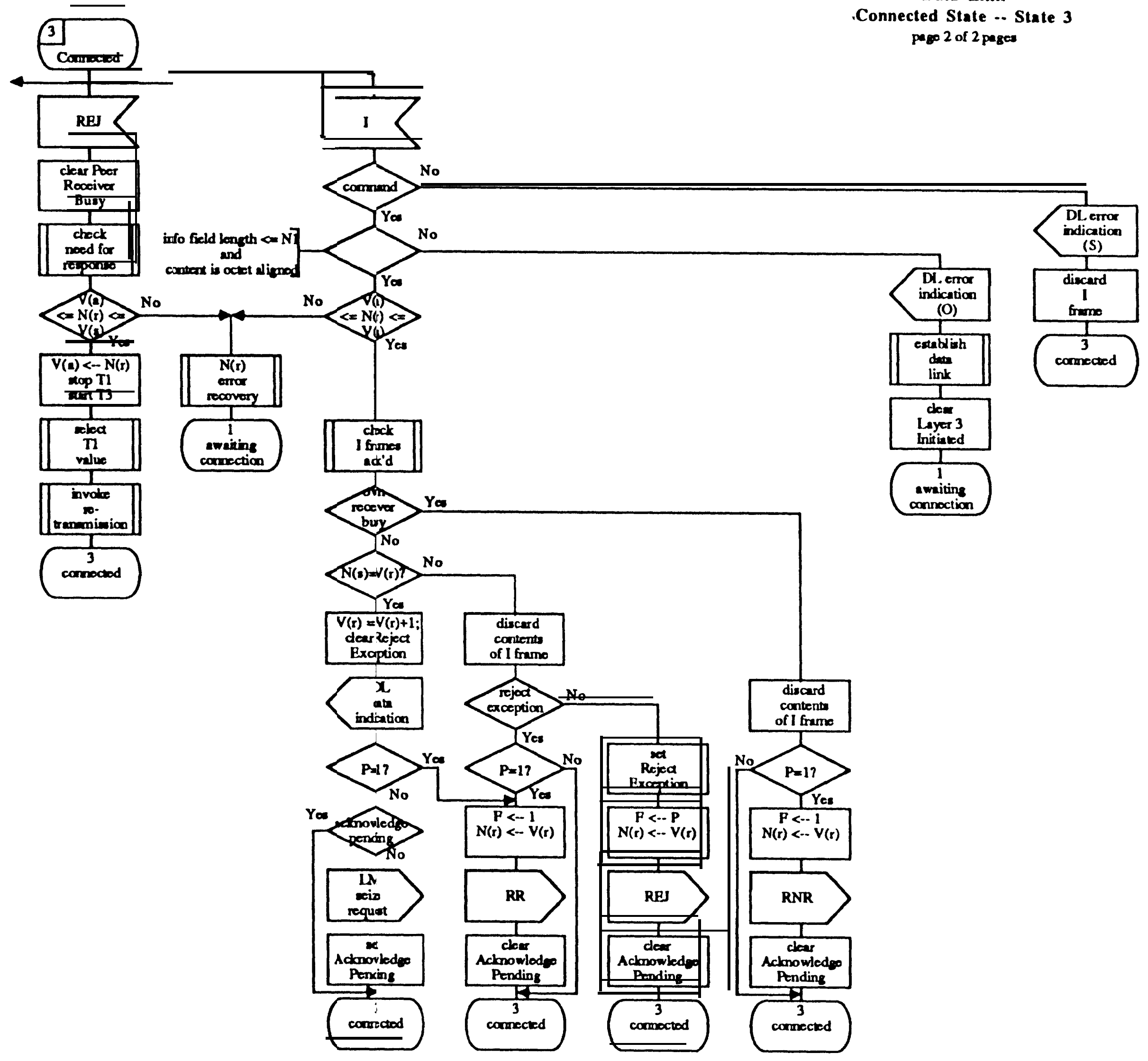
A -- F=1 received but P=1 not outstanding.
B -- Unexpected DM with F=1 in states 3, 4, and 5.
C -- Unexpected UA in states 3, 4, and 5.
D -- UA received without F=1 when SABM or DISC was sent with P=1.
E -- DM received in states 3, 4, or 5.
F -- Data link reset; i.e., SABM received in state 3 or 4.
I -- N2 timeouts: unacknowledged data.
J -- N(R) sequence error.
L -- control field invalid or not implemented.
M -- information field was received in a U- or S-type frame.
N -- length of frame incorrect for frame type.
O -- I-frame exceeded maximum allowed length.
P -- N(s) out of the window.
Q -- UI response received, or UI command with P=1 received.
R -- UI frame exceeded maximum allowed length.
S -- I response received.
T -- N2 timeouts: no response to enquiry.
U -- N2 timeouts: extended peer busy condition.

Flags & Parameters

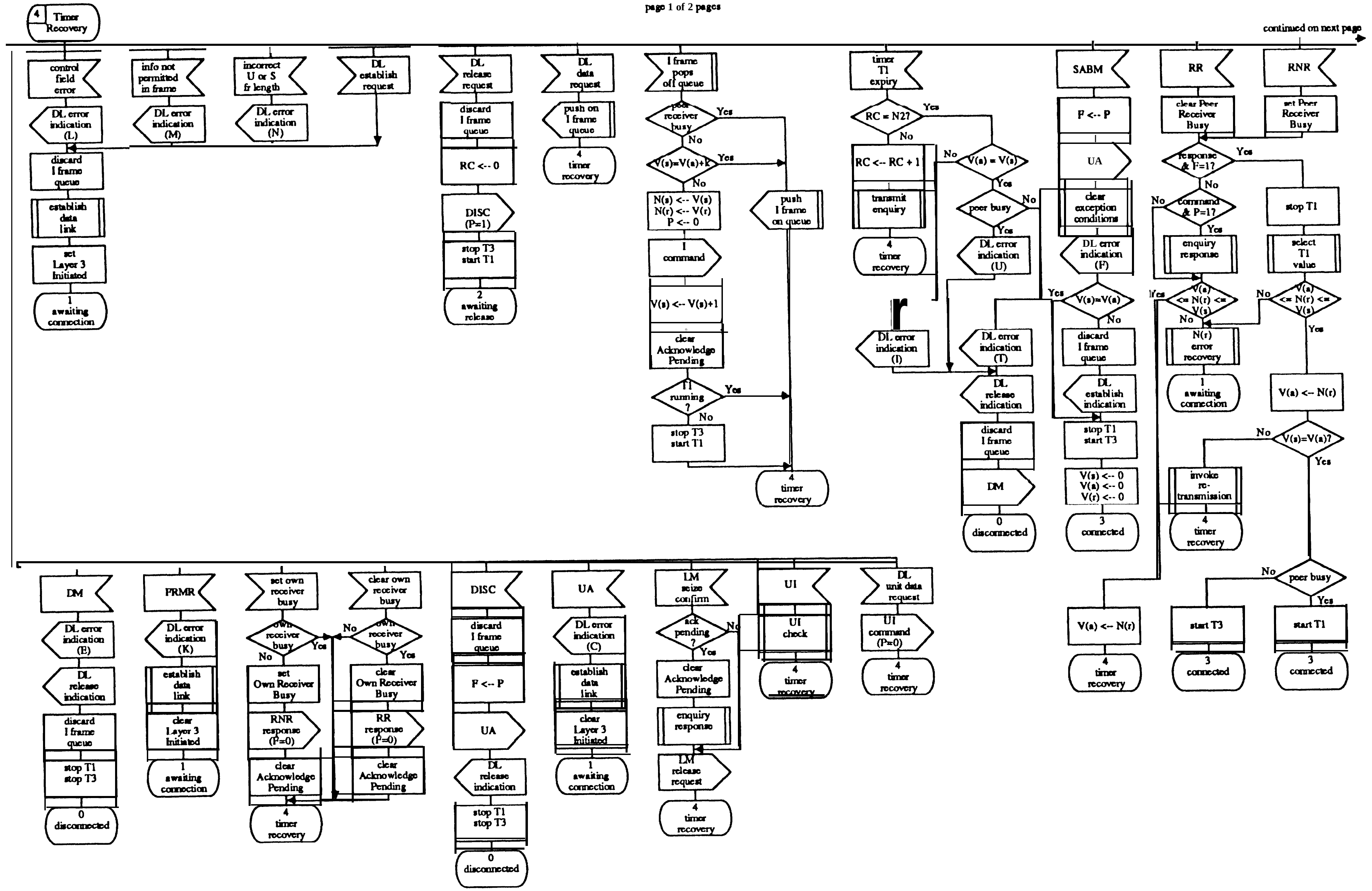
Layer 3 Initiated -- SABM was sent by request of Layer 3; i.e., DL-Establish-Request primitive.
Peer Receiver Busy -- remote station is busy and can not receive I-frames.
Own Receiver Busy -- Layer 3 is busy and can not receive I-frames.
Reject Exception -- a REJect frame has been sent to the remote station.
Acknowledge Pending -- I-frames have been successfully received but not yet acknowledged to the remote station.
SRT -- smoothed round trip time.
T1V -- next value for T1; default initial value is initial value of SRT.
N1 -- maximum number of octets in the information field of a frame, excluding inserted 0-bits.
N2 -- maximum number of retries permitted.

rev 88 Sep 30 K3NA

Data Link
·Connected State -- State 3
page 2 of 2 pages

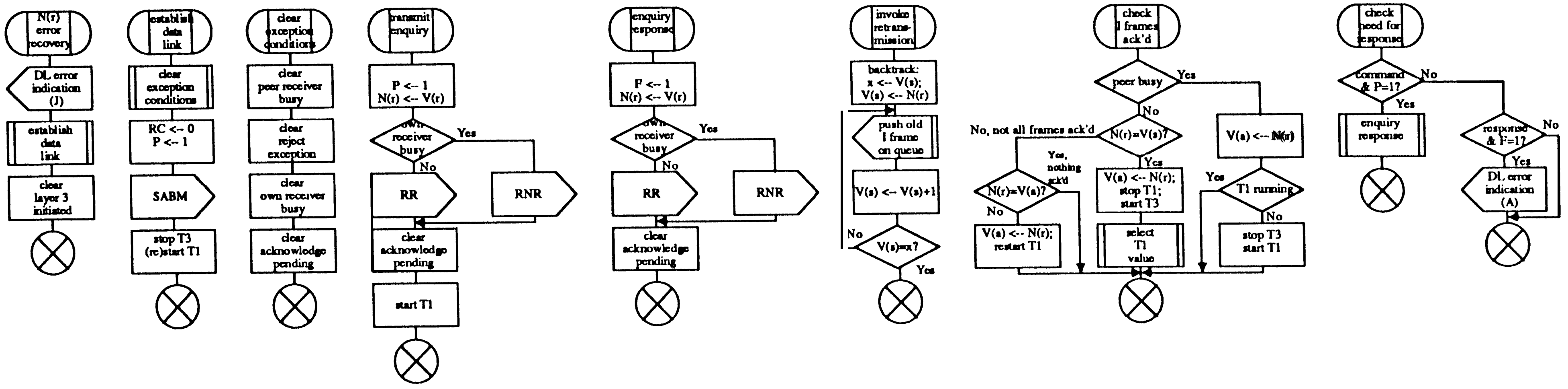


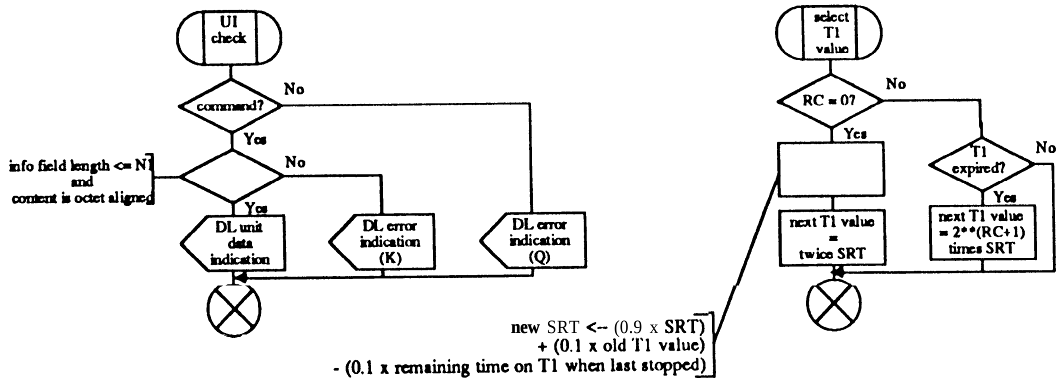
Data Link
Timer Recovery State -- State 4
page 1 of 2 pages



rev 8% Sep 30 K3NA

Data Link
Subroutines





Parameter Negotiation between Consenting AX.25 Stations

Eric L. Scace K3NA
10701 Five Forks Road
Frederick MD 21701
USA

0. Summary

This paper is part of a series of papers which introduce enhancements and other improvements in AX.25. This particular paper provides a mechanism for the negotiation of connection parameters on an AX.25 link between two consenting stations and the affected digipeaters. The negotiation mechanism is based on existing provisions for link parameter notification and negotiation which are found in a number of international telecommunications standards (ISO and CCITT). These procedures are backwards compatible with existing AX.25 Revision 2.0 implementations; and may also be expanded to include the notification or negotiation of other aspects of link operation.

1. Status of Proposal

The ARRL Digital Committee specifically directed me to develop a proposal for parameter negotiation mechanisms to provide for automatic adjustment of the following aspects of an AX.25 link between two stations:

a) maximum frame size (N1) and window size.
b) round trip timers.
c) physical transmission speed.
d) segmentation procedures.

The following material is still in a draft state. You are invited to review and comment on this material. Comments are desired so that the final publication is as useful as possible to its readers.

2. Notification versus Negotiation

Notification and negotiation are related but distinct concepts.

Notification occurs when one station provides to another some information about its present operating parameters. The operating parameters of the sending station are *not* changed; the information provided is considered advisory in nature. The receiving station may, however, elect to adjust its internal operating parameters based on the contents of the notification.

For example, notification could be used to inform the remote station that, henceforth, all future UI and I frames will contain up to 4096 octets of data in the information field. The remote station may wish to adjust its internal buffer allocation procedures accordingly when such a notice is received.

Negotiation occurs when stations exchange information about operating parameters with the intention to reach a mutually-agreed value. The operating parameters of both stations may be changed as a result of the negotiation. Generally, negotiated parameters have a *default value* towards which negotiation is made.

For example, negotiation could begin with one station indicating that it wishes to send UI and I frames which contain up to 4096 octets of data in the information field. The default value in AX.25 today is 256 octets. The remote station (or an intervening digipeater) may indicate that only 1024 octets can be accomodated, with the final result that both stations settle on the 1024 octet value. Here, negotiation took place from the initial proposed value towards the default value.

The choice of permitting negotiation or notification should be made on a parameter by parameter basis; some parameters are better negotiated, whilst others are more satisfactorily handled by simple notification.

There are also some parameters which *are* best left unnegotiated; for example, it is not appropriate to allow one station to change the callsign stored in the TNC of another station.

3. Outline of Procedure

The proposed procedure utilizes standard XID (Exchange Identification) command and response frames. The XID frame is defined in national and international standards for high-level data link control (HDLC) protocols.

In the architecture of extended finite state machines described in other companion papers, this procedure is executed by a new state machine, the data link management entity. Figure 1 illustrates the relationship between this state machine and other state machines associated with AX.25. Note that, for a pure digipeater, a data link management state machine exists even though no instances of data link state machines or AX.25 users may exist within the pure digipeater.

Notification/negotiation may occur at any time during the life of the link to another station.

Notification/negotiation affect only the parameters used on a particular link between two particular stations. System-wide default parameters within a TNC should not be changed by a remote station through this procedure, nor should a remote station change parameters associated with links to other (third-party) stations.

3.1 Procedure Between Stations Equipped for XID

Notification/negotiation begins with the AX.25 user sending an **MDL Negotiate Request** primitive to the data link management state machine. (MDL stands for Management -- Data Link.) The data link management state machine transmits an XID command frame with the Poll bit set to '1'. Timer TM201 supervises the procedure. The transmitting data link state machine enters state 1, negotiating.

The remote station's data link management state machine processes the contents of the received XID command, and then replies with an XID response frame with the Final bit set to '1'. State 0, ready, is then re-entered. The local AX.25 users are not notified, since a connection to user may not exist at this point in time.

Upon receipt of the XID response the initiating state machine stops timer TM201, delivers an **MDL Negotiate Confirm** primitive to the AX.25 user, and returns to state 0, ready.

If timer TM201 expires, the XID command is retransmitted. NM201 attempts are made before the notification/negotiation procedure is abandoned and an MDL **Error** primitive is delivered to the AX.25 user.

3.2 Processing of XID by Digipeaters

Intervening digipeaters must examine the contents of the XID to determine if the parameters being adjusted are compatible with their capabilities and their responsibilities; for example, under this proposal a digipeater would refuse to permit a change in operating speed since, as a result of making the change, the digipeater becomes unavailable to other stations still using the original speed. This examination is carried out by the data link management state machine of the digipeater.

In order to positively indicate that the XID frame has been examined by each intervening digipeater, the digipeater inserts its callsign into the information field of the XID frame. The destination station which receives the XID *command* must check to see if all intervening digipeaters have performed their compatibility checks. If one or more digipeater callsigns are absent within the information field of the XID frame, then two possibilities exist:

a) the negotiation/notification is invalid, since intervening digipeaters did not consent to or recognize (i.e., not an AX.25 Revision 2.1 implementation) the procedure. Appropriate action to abandon the notification/negotiation procedure is taken by each station.

b) the negotiation/notification procedure is valid, because the parameter being negotiated does not affect digipeater operation.

Whether (a) or (b) should be followed is indicated below for each specific notification/negotiation procedure.

3.3 Backwards Compatibility

If an existing AX.25 Revision 2.0 station receives an XID command, it will either:

a) if in the disconnected state, ignore the frame; or,

b) if in the connected state, transmit an FRMR response.

The station initiating notification/negotiation will detect case (a) through T1 timeout, and can detect case (b) by inspection of the FRMR response. In both cases the MDL Error primitive is returned to the AX.25 user and default parameters are used.

An existing AX.25 Revision 2.0 digipeater will digipeat the XID frames blindly without checking the contents for compatibility. The mechanism for insuring that each digipeater has checked in §

4. Format of XID Frames

The XID frame is a U-type frame, and contains an information field.

4.1 XID Control Field

The XID frame control field contains:

8	7	6	5	4	3	2	1	<-- transmitted bit number
1	0	1	**p/f**	1	1	1	1	

4.2 Organization of the Information Field Within the XID Frame

The XID information field is organized in a modular fashion, following the structuring rules adopted by the CCITT in Recommendation Q.931.

The basic building block is the ***information element*** which contains related information about a particular parameter being notified or negotiated. Information elements may be included in any order within the information field. Only the information elements relevant to the parameters being notified or negotiated are included in a particular XID frame.

Each information element is comprised of three parts:

a) the information element identifier, a single octet whose value indicates which information element follows.

b) the length, a single octet containing a binary number. The binary number represents the length, in octets, of the remaining contents of the information element; i.e., excluding the information element identifier and length fields.

c) the contents, containing one or more fields in an integral number of octets. The exact organization depends on the specific information element, as diagrammed below.

All three parts are transmitted together as an indivisible unit. An information element may be repeated, if appropriate.

Important flexibilities are gained by the inclusion of the length field. The contents of the information element may be variable in length, and can even be expanded over time as a result of subsequent enhancements. Furthermore, a particular implementation need not implement every possible information element. In the event that an unimplemented information element is received (i.e., the value of the information element identifier is not recognized), the receiver can ignore that element and, by checking the length field, skip down to the next information element.

4.3 Digipeater Compatible

This information element is used to inform the XID receiver that the indicated digipeater has examined *the* contents of *the* XID *command* and has determined that it is compatible with the parameter(s) as presently indicated.

All AX.25 Revision 2.1 digipeaters will examine the contents of each information element in any XID command to be digipeated. Those information element which were understood (i.e., the information element identifier was recognized by the digipeater implementation) will be processed according to the specifications in § 5 below. The digipeater shall add a Digipeater Compatibility Information Element, containing the digipeater callsign, to the digipeated XID command. Within the Digipeater Compatibility Information Element, bit flags are set indicating which information elements are implemented by the digipeater. This permits "forward compatibility", whereupon, in the future, additional information elements can be included without:

- affecting previous digipeater implementations
- confusing the remote station on the link as to whether all information elements were understood by all intervening equipments.

Existing AX.25 Revision 2.0 digipeaters will digipeat ***the MD*** command ***without*** including a Digipeater Compatible Information Element containing its callsign, since they do not presently understand this frame.

The Digipeater Compatible Information Element is not included in XID *responses*.

The format of the Digipeater Compatible Information Element is shown in Figure 2.

4.4 N1

This information element is used to negotiate the value of data link parameter N1, maximum information field length. The format of the N1 Information Element is shown in Figure 3.

4.5 Window Size

This information element is used to negotiate the value of the data link parameter k, window size. The format of the Window Size Information Element is shown in Figure 4.

4.6 Initial Round Trip Time

This information element is used to notify the remote station of the initial value to be used in calculating round trip time. The format of the Initial Round Trip Time Information Element is shown in Figure 5.

4.7 Transmission Speed

This information element is used to negotiate the transmission speed to be used on the link. The format of the Transmission Speed Information Element is shown in Figure 6.

4.8 Average Information Transfer Rate

This information element is used to notify the remote station of the average information transfer rate which is expected to occur on the connection. The format of the Average Information Transfer Rate Information Element is shown in Figure 7.

4.9 Changeover

This information element is used to coordinate the synchronized change from one data link operating condition to another. The changeover procedures are general purpose in that they can be employed for a variety for data link changes in the future. However, at the present time only the transmission speed negotiation procedures employ a synchronized changeover. Note that, because of the inability of digipeaters to guarantee end-to-end delivery of XID frames, synchronized changeover is not permitted on links containing digipeaters. The format of the Changeover Information Element is shown in Figure 8.

5. Notification/Negotiation Procedures for Specific Parameters

This section describes how each station evaluates and responds to notification or negotiation of the various data link parameters.

5.1 Segmentation Availability Notification

The segmentation procedures (described in a companion paper) are presently proposed as an inherent part of the AX.25 Revision 2.1 protocols, as are these general procedures for notification/negotiation. Any station which sends an XID frame is, by definition, operating according to the proposed AX.25 Revision 2.1 and therefore can also employ the segmentation procedures.

Therefore, to notify a remote station that the segmentation procedures (and other aspects of AX.25 Revision 2.1) are available for use, the data link management state machine shall cause an XID frame to be transmitted to the remote station.

No information element is required in the XID frame to notify the remote station that segmentation and other AX.25 Revision 2.1 enhancements are available.

Intervening digipeaters are transparent to segmentation procedures. Thus, the absence of one or more intervening digipeater identifications (detected by comparing the Digipeater Compatible Information Elements within the XID frame with the digipeater callsigns in the XID frame's address field) does not preclude the use of segmentation procedures; i.e., § 3.2 case (b) applies.

5.2 N1 Negotiation

Parameter N1 is the maximum number of octets which may be contained in the information field of an I, UI or XID frame; the flags, address fields, frame check sequence, and 0-bits added for transparency are *not* included in calculating this limit. The N1 value applies to both directions of information transfer.

The default value of N1 is 256 octets. N1 negotiation procedures are used when one station wishes to arrange with a remote station to use a larger or smaller value of N1. Permissible values of N1 are powers of 2; e.g., 32, 64, 128, 256, 512, 1024, 2048, and 4096 are all examples of permissible values of N1.

N1 negotiation procedures begin with the transmission of an XID command with the N1 Information Element containing the proposed new value of N1. Note that, at the time of transmission of the XID command, the new value of N1 *is not yet authorized for use.* Any ongoing link operations shall continue to use the present (old) value of N1 for the time being.

Since the combination of N1 and window size affects the buffer requirements of intervening digipeaters, the data link management state machine within each digipeater on the path must evaluate the proposed N1. If the proposed N1 is not acceptable to the digipeater, the digipeater shall replace the proposed value with a new, *smaller,* proposed value. For example, if a value of 4096 octets was proposed, an intervening digipeater may trim down the value to 1024 octets... or even all the way back to 16 octets (which might occur if a two-port digipeater which relayed from an excellent radio path to a poor radio path was involved).

The remote station receiving the XID command first compares the digipeater address fields with the Digipeater

Compatibility Information Elements. If one or more digipeaters failed to include a Digipeater Compatibility Information Element, or if *one* or more digipeaters indicate that the N1 Information Element was *not* evaluated, the station concludes that there are some incompatible digipeaters in the path, and that N1 negotiation can not be completed. The XID response frame is transmitted with the N1 Information Element coded as "default N1 must be used".

If all Digipeater Compatibility Information Elements are present, the station receiving the XID command then evaluates the proposed value for N1 and either accepts it or selects a new smaller value. At this point the final value for N1 has been obtained; this final value is recorded in the N1 Information Element contained in the XID response.

As the XID response works its way back to across the link, intervening digipeaters and the station initiating N1 negotiation shall note the final value chosen for N1 and plan accordingly.

5.3 Window Size Negotiation

Parameter k is *the* maximum number of I frames which may be outstanding (i.e., transmitted but not yet acknowledged) on the data link. The k value is independent for each direction of data transfer on the link.

The default value of k is seven I frames. Window size negotiation procedures are used when one station wishes the remote station to prepare for larger or small quantities of outstanding frames. Any integer in the range from one to seven is permitted.

Note -- The use of window sizes larger than seven requires extended sequence numbering and the SABME command. The ARRL Digital Committee has not yet discussed whether extended sequence numbering should be a part of AX.25 Revision 2.1.

Window size negotiation procedures begin with the transmission of an XID command with the Window Size Information Element containing the proposed new value of k. Note that, at the time of transmission of the XID command, *the* new value of k *is not yet authorized for use. Any* ongoing link operations shall continue to use the present (old) value of k for the time being.

Since the combination of N1 and k affects the buffer requirements of intervening digipeaters, the data link management state machine within each digipeater on the path must evaluate the proposed k. If the proposed k is not acceptable to the digipeater, the digipeater shall replace the proposed value with a new, *smaller,* proposed value. For example, if a value of 6 was proposed, an intervening digipeater may trim down the value to 4 or even all the way to 1.

The remote station receiving the XID command first compares the digipeater address fields with the Digipeater Compatible Information Elements. If one or more digipeaters failed to include a Digipeater Compatible Information Element, or if one or more digipeaters indicated that the Window Size Information Element was *not* evaluated, the station concludes that *there* are some incompatible digipeaters in the path, and that window size negotiation can not be completed. The XID response frame is transmitted with the Window Size Information Element coded as "default k must be used".

If all Digipeater Compatibility Information Elements are present, the station receiving the XID command then evaluates the proposed value for k and either accepts it or selects a new smaller value. At this point the final value for k has been obtained; this final value is recorded in the Window Size Information Element contained in the XID response.

As the XID response works its way back to across the link, intervening digipeaters and the station initiating window size negotiation shall note the final value chosen for k and plan accordingly.

5.4 Initial Round Trip Time Notification

The initial round trip time notification procedures allow one station to suggest to another an appropriate initial value for the smoothed round trip time calculations. The suggestion could be based, for instance, on the present smoothed round trip times seen on other data links over the same radio path, or may be completely arbitrary. The value of smoothed round trip time is ultimately calculated independently by each station on the link, based on recent historical trends for acknowledgements.

Initial round trip time notification procedures begin with the transmission of an XID command with the Initial Round Trip Time Information Element containing the new value for smoothed round trip time to be used by the remote station.

Since round trip timing is not done by digipeaters, each digipeater takes *no* action on this information element.

The remote station shall substitute the round trip time value contained in the XID command in place of its present value in the data link state machine. An XID response must be transmitted (to stop timer TM201); if no other items are being negotiated, the XID response frame may wind up containing an empty information field.

The absence of one or more intervening digipeater identifications (detected by comparing the Digipeater Compatible

Information Elements within the XID frame with the digipeater callsigns in the XID frame's address field) does not preclude the use of initial round trip time notification procedures; i.e., § 3.2 case (b) applies.

5.5 Transmission Speed Negotiation

AX.25 has no default value for transmission speed, although 1200 bit/s and 300 bit/s are commonly used on VHF and HF, respectively. Transmission speed negotiation procedures are used when one station wishes to execute a synchronized change to a new (faster or slower) transmission speed.

In order to achieve a fully synchronized and reliable changeover from one speed to another, a two phase procedure is employed. In the first phase, negotiation of the new speed is performed. In the second phase, the stations execute the changeover to the new speed. The use of both phases is recommended but not mandatory; if ***apriori*** knowledge exists that all stations can operate at the new speed, the negotiation phase may be omitted.

Note -- This proposal does not support the alteration of the transmission speed of digipeaters. Changing the transmission speed of a digipeater would remove the digipeater from service for other users. Thus, transmission speed negotiation can be executed only if no digipeaters are part of the link.

5.5.1 Negotiation

Transmission speed negotiation procedures begin with the transmission of an XID command with the Transmission Speed Information Element containing a list of one or more transmission speeds, in order of preference. Note that, at the ***time*** of transmission of ***this*** first XID ***command,*** a new transmission ***speed is not yet authorized for use. Any ongoing*** link operations shall continue to use the present transmission speed for the time being.

Since any possible speed change will affect the ability to use intervening digipeaters, the data link management state machine within each digipeater on the path checks to determine if transmission speed negotiation is being attempted. If an attempt is being made to negotiate transmission speed, the digipeater alters the Transmission Speed Information Element to "present speed must be used".

The remote station receiving the XID command should also check to see if transmission speed negotiation is being attempted over a link with digipeaters. If so, the XID response frame is transmitted with the Transmission Speed Information Element coded as "present speed must be used".

If no digipeaters are included in the link, the station receiving the XID command then evaluates the proposed values for transmission speed and selects the first one in the list which it can support. The selected value is reported back across the link in the Transmission Speed Information Element contained in the XID response. No change in transmission speed occurs at this point, however.

When the XID response is received at the initiating station, the execution of the changeover begins.

5.5.2 Execution of Changeover

Upon completion of successful transmission speed selection, a new XID command is transmitted at the old speed by the initiating station, containing *only* two information elements:

- the Transmission Speed Information Element, coded with the selected new speed, and
- the Execute Changeover Information Element, coded to indicate "execute changeover".

The initiating station starts TM202 and enters state 2: changeover. Immediately after the XID command has been sent, the station shifts to the new speed to await the XID response. Note -- it is desirable, but not mandatory, that the station also continue to monitor at the old speed for possible indication of changeover failure, and to receive other frames from the remoton until changeover completes.

An intervening digipeater (if any) which detects an XID frame (command or response) containing the Changeover Information Element shall alter the coding of that information element to read "changeover failure".

The remote station, upon detecting the Changeover Information Element coded "execute changeover", shall understand that a changeover is in progress. The Transmission Speed Information Element shall be examined to determine if the new speed can be supported. If the new speed can be supported, the station shall immediately shift to the new speed and shall transmit an XID response with the Changeover Information Element coded "changeover confirmed". If the new speed can not be supported, the station shall transmit an XID response containing the Changeover Information Element coded "changeover failure".

On the other hand, if the remote station detects the Changeover Information Element coded "changeover failure", it shall understand that an attempt was made to made a non-negotiated changeover which could not be supported by one or more intermediate digipeaters. The remote station shall transmit an XID response frame containing the Changeover Information Element as it was coded in the XID command.

The initiating station, upon receipt of an XID response containing the Changeover Information Element coded "changeover confirmed", shall stop timer TM202; shall send an MDL Neogitate Confirm primitive to the AX.25 user; and shall enter state 0: ready. The changeover procedures shall be considered completed.

The initiating station, upon receipt of an XID response containing the Changeover Information Element coded "changeover failure", shall stop timer TM202; shall revert back to the previous speed; shall send a MDL Error primitive to the AX.25 user; and shall enter state 0: ready.

If timer TM202 expires the initiating station shall conclude that the changeover failed; shall revert back to the previous speed; shall send a MDL Error primitive to the AX.25 user; and shall enter state 0: ready.

5.6 Average Information Transfer Rate Notification

The average information transfer rate notification procedures allow one station to inform the other of the anticipated long-term average information transfer rate; such averages can sometimes help in determining whether sufficient processor respources remain available to offer the human user reasonable response time to requests. The average information transfer rate is independently established for each direction of data transfer.

Average information transfer rate notification procedures begin with the transmission of an XID command with the Average Information Transfer Rate Information Element containing the new value for estimated average information transfer rate to be used by the remote station for planning purposes.

Since this average does not affect digipeaters, each digipeater takes no action on this information element.

Upon receipt of the XID command, the remote station shall transmit an XID response; the XID response may also contain an Average Information Transfer Rate Information Element, indicating the anticipated average information transfer rate in the reverse direction on the data link.

The absence of one or more intervening digipeater identifications (detected by comparing the Digipeater Compatible Information Elements within the XID frame with the digipeater callsigns in the XID frame's address field) does not preclude the use of the average information transfer rate notification procedures; i.e., § 3.2 case (b) applies.

6. Combinations of Procedures

Simultaneous negotiation and notification of various parameters is permitted and would be conducted by including all of the relevant information elements in a single XID frame. Figure 9 contains an example of an XID frame.

Changeover procedures must be conducted independently of negotiation/notification procedures. Although the changeover procedures are only exercised by the transmission speed change described above, other synchronized changeovers could be envisioned (transmitter frequency, etc) and could be combined together into a single XID frame.

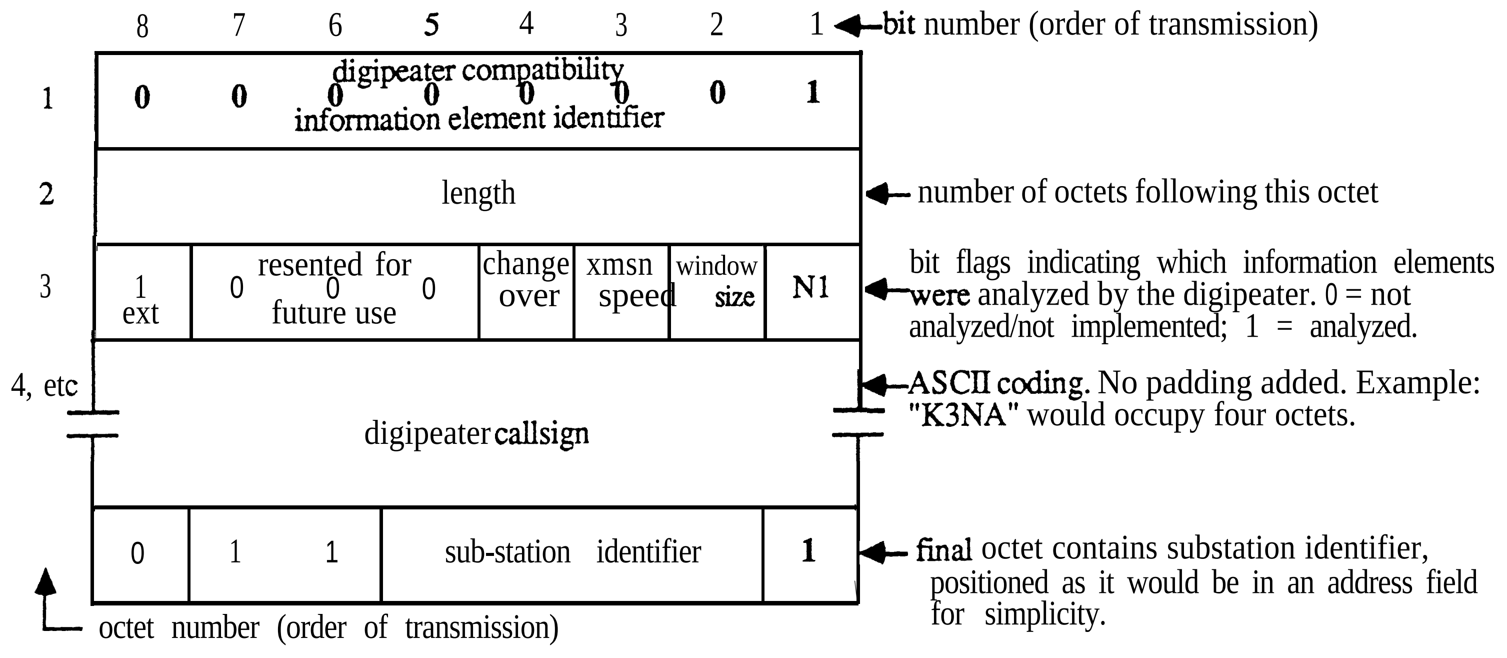
7. SDL Representation of Procedures

Attached at the end of this paper are two sets of SDL diagrams. One set defines the negotiation, notification, and changeover procedures for the stations at each end of an AX.25 data link. The second set defines the negotiation and changeover procedures for intervening digipeaters on the AX.25 data link.

8. Summary

A general purpose, flexible mechanism for automatically adjusting various link parameters, and for conducting a synchronized change of transmission speed, has been proposed. The structure and format of these procedures are both backwards compatible, and yet able to accomodate further expanded applications which might be developed in the future.

Your comments and suggestions are welcome.


8 7 6 5 4 3 2 1
bit number (order of transmission)
1
0 0 0 0 0 0 0 1
digipeater compatibility information element identifier
2
length
number of octets following this octet
3
1 ext
0 0 0
resented for future use
change over
xmsn speed
window size
N1
bit flags indicating which information elements were analyzed by the digipeater. 0 = not analyzed/not implemented; 1 = analyzed.
4, etc
digipeater callsign
ASCII coding. No padding added. Example: "K3NA" would occupy four octets.
0 1 1
sub-station identifier
1
final octet contains substation identifier, positioned as it would be in an address field for simplicity.
octet number (order of transmission)

Note -- Octer 3 bit 8 is an extension bit provided for future expansion. Therefore all equipment must interprete this bit, when set to 1, to indicate that this is the last octet of bit flags. When set to 0, this bit indicates that another octet of bit flags follows.

Figure 2 -- Digipeater Compatibility information element

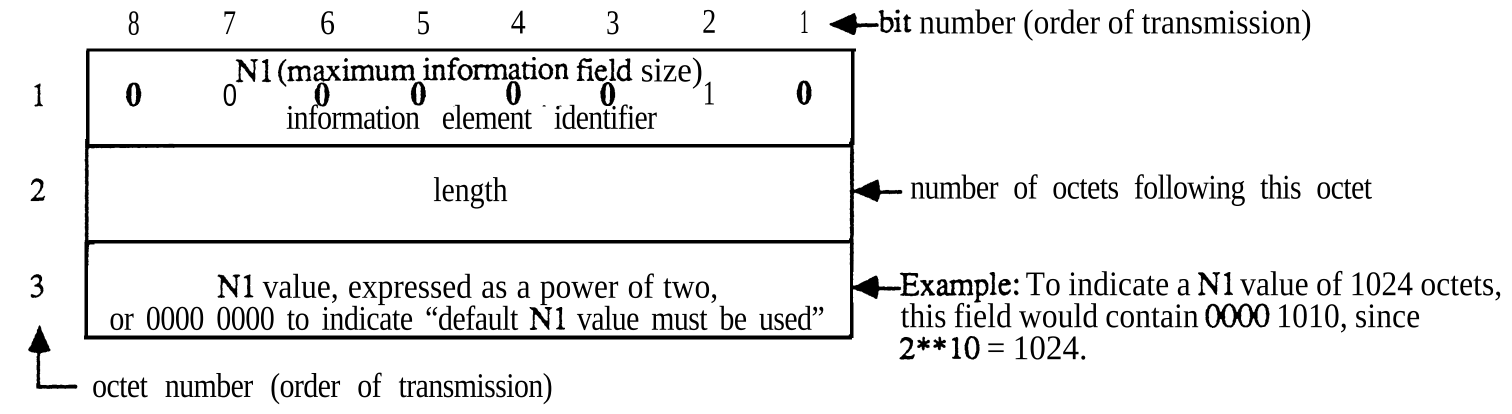

8 7 6 5 4 3 2 1
bit number (order of transmission)
1
0 0 0 0 0 0 1 0
N1 (maximum information field size) information element identifier
2
length
number of octets following this octet
3
N1 value, expressed as a power of two, or 0000 0000 to indicate "default N1 value must be used"
Example: To indicate a N1 value of 1024 octets, this field would contain 0000 1010, since 2**10 = 1024.
octet number (order of transmission)

Figure 3 -- Nl information element

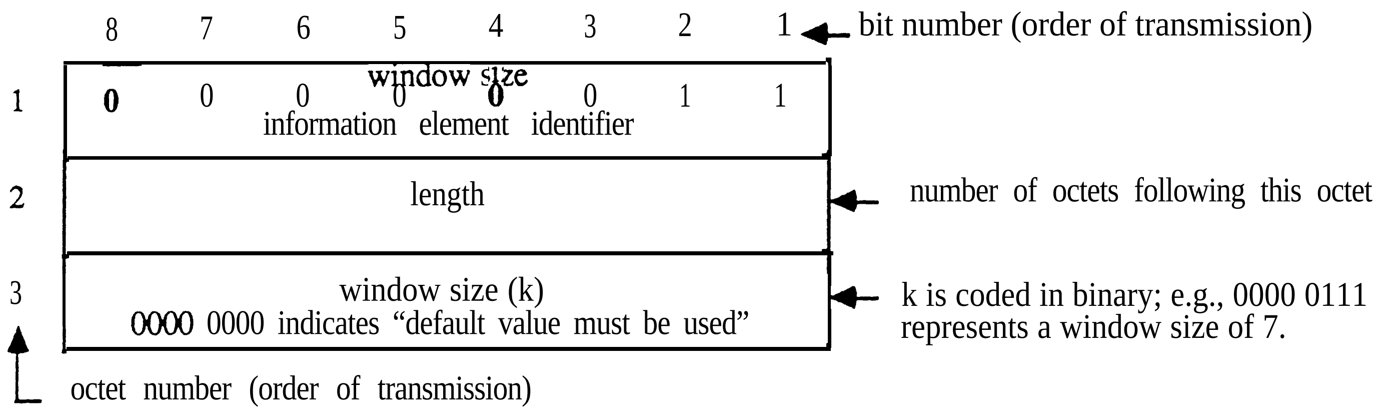

8 7 6 5 4 3 2 1
bit number (order of transmission)
1
window size
0 0 0 0 0 0 1 1
information element identifier
2
length
number of octets following this octet
3
window size (k)
0000 0000 indicates “default value must be used”
k is coded in binary; e.g., 0000 0111 represents a window size of 7.
octet number (order of transmission)

Figure 4 -- Window Size Information Element

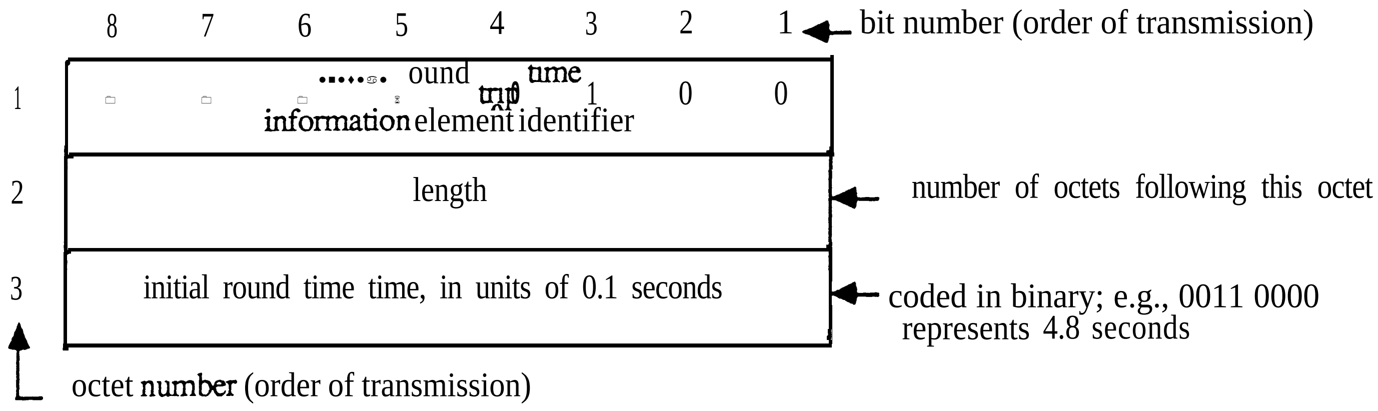

8 7 6 5 4 3 2 1
bit number (order of transmission)
1
ound trip time
0 1 0 0
information element identifier
2
length
number of octets following this octet
3
initial round time time, in units of 0.1 seconds
coded in binary; e.g., 0011 0000 represents 4.8 seconds
octet number (order of transmission)

Figure 5 -- Initial Round Trip Time Information Element

	8	7	6	5	4	3	2	1 ← bit number (order of transmission)	
1	0	0	0	0	0	1	0	1	transmission speed information element identifier
2	length								← number of octets following this octet
3	1 ext	0	0	transmission speed coded according to the following table					← transmission speed encodings:

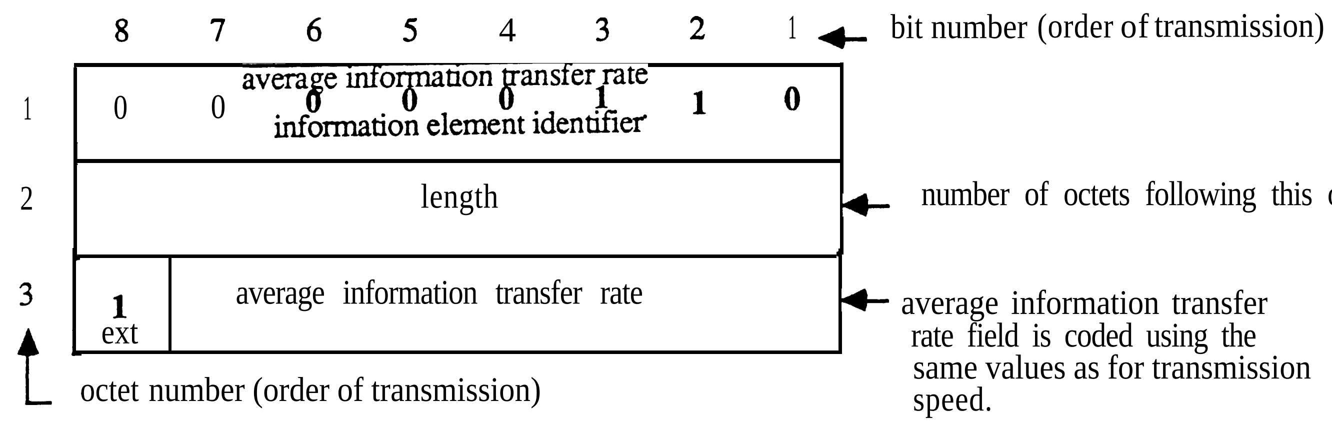
↑ octet number (order of transmission)

0 0000	-- present speed must be used.
0 0001	-- 0.6 kbit/s
0 0010	-- 1.2 kbit/s
0 0011	-- 2.4 kbit/s
0 0100	-- 3.6 kbit/s
0 0101	-- 4.8 kbit/s
0 0110	-- 7.2 kbit/s
0 0111	-- 8 kbit/s
0 1000	-- 9.6 kbit/s
0 1001	-- 14.4 kbit/s
0 1010	-- 16 kbit/s
0 1011	-- 19.2 kbit/s
0 1100	--32 kbit/s
0 1110	-- 38 kbit/s
0 1111	-- 56 kbit/s
10000	-- 64 kbit/s
10011	-- 384 kbit/s
1 0101	-- 1536 kbit/s
10111	-- 1920 kbit/s
1 1001	-- 0.050 kbit/s
1 1010	-- 0.074 kbit/s
1 1011	-- 0.110 kbit/s
1 1100	-- 0.150 kbit/s
1 1101	-- 0.2 kbit/s
1 1110	-- 0.3 kbit/s
1 1111	-- 12 kbit/s

Note 1 -- Octet 3, bit 8 is reserved for use as an extension bit. Since all transmission speed encodings presently fit into one octet, this bit (for the time being) will always be set to "1".

Note 2 -- When multiple transmission speeds are to be listed octet 3 shall be repeated. The first instance of octet 3 shall be considered to be the most preferred transmission speed; the second instance shall be considered the second-choice speed; etc.

Figure 6 -- Transmission Speed Information Element


8 7 6 5 4 3 2 1
bit number (order of transmission)
1
0 0 0 0 0 1 1 0
average information transfer rate
information element identifier
2
length
number of octets following this octet
3
1
ext
average information transfer rate
average information transfer rate field is coded using the same values as for transmission speed.
octet number (order of transmission)

Note -- See the transmission speed information element for a description on the use of octet 3, bit 8.

Figure 7 -- Average Information Transfer Rate Information Element

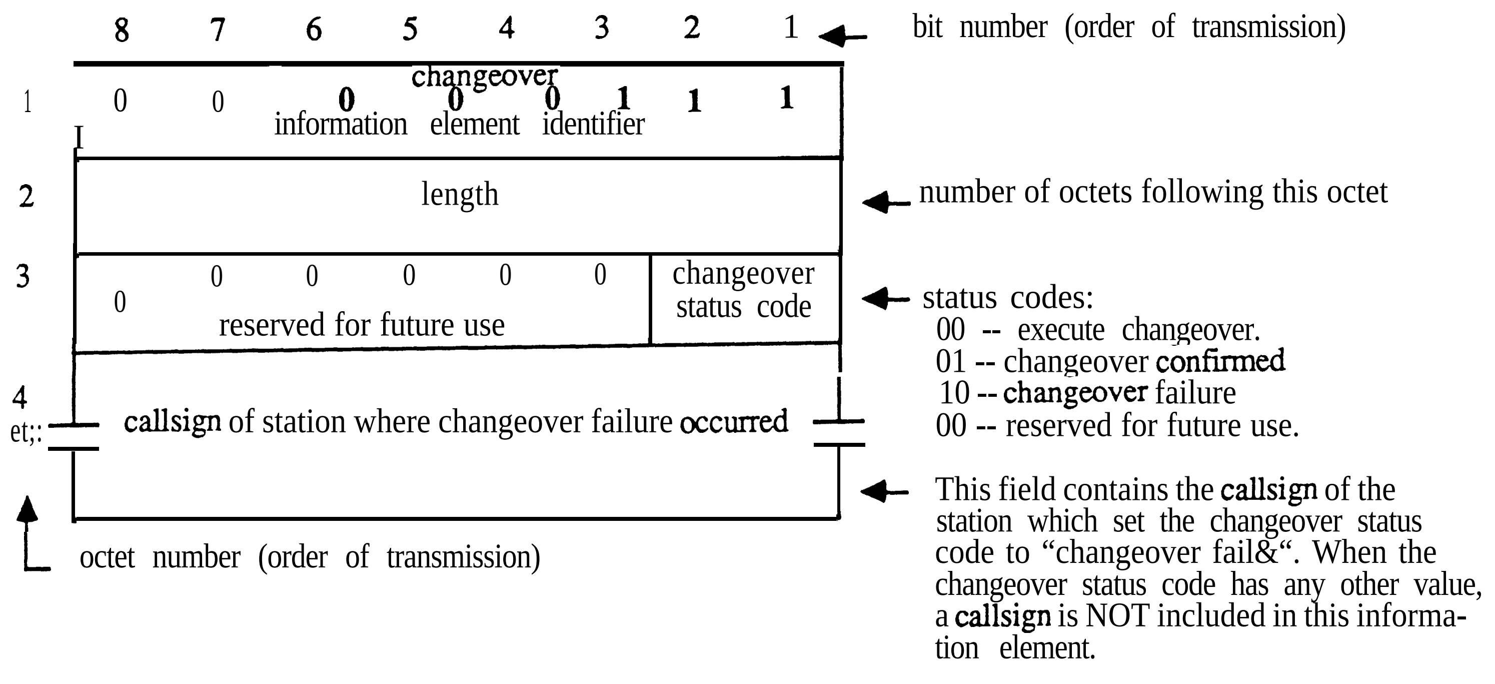

8 7 6 5 4 3 2 1
bit number (order of transmission)
1
0 0 0 0 0 1 1 1
changeover
information element identifier
2
length
number of octets following this octet
3
0 0 0 0 0 0
reserved for future use
changeover status code
status codes:
00 -- execute changeover.
01 -- changeover confirmed
10 -- changeover failure
00 -- reserved for future use.
4
et;:
callsign of station where changeover failure occurred
This field contains the callsign of the station which set the changeover status code to “changeover fail&“. When the changeover status code has any other value, a callsign is NOT included in this information element.
octet number (order of transmission)

Figure 8 -- Changeover Information Element

8	7	6	5	4	3	2	1	bit order of transmission
(address fields)								
1	0	1	1	1	1	1	1	**XID control field; P/F bit = 1.**
0	0	0	0	0	0	1	0	**Nl information element.**
0	0	0	0	0	0	0	1	one octet follows this octet.
0	0	0	0	1	0	1	0	Nl value = 1024 octets.
0	0	0	0	0	0	1	1	**window size information element.**
0	0	0	0	0	0	0	1	one octet follows this octet.
0	0	0	0‘	0	1	1	1	k = 7.
0	0	0	0	0	1	0	0	**inital round trip time info element**
0	0	0	0	0	0	0	1	one octet follows this octet
0	0	1	1	0	0	0	0	round trip time = 4.8 seconds
0	0	0	0	0	1	0	1	**transmission speed info element**
0	0	0	0	0	0	1	1	three octets follow this octet
1	0	0	0	1	1	1	1	56 kbit/s is preferred speed.
1	0	0	0	0	0	1	1	2.4 kbit/s is second choice.
1	0	0	0	0	0	1	0	1.2 kbit/s is third choice.
0	0	0	0	0	0	0	1	**digipeater compatibility info element**
0	0	0	0	0	1	1	0	six octets follow this octet.
1	0	0	0	0	0	1	1	only window size and N1 was understood and analyzed
0	1	0	0	1	0	1	1	K3NA digipeater has checked for compatibility.
0	0	1	1	0	0	1	1	
0	1	0	0	1	1	1	0	
0	1	0	0	0	0	0	1	
0	1	1	0	0	0	0	1	
0	0	0	0	0	0	0	1	**digipeater compatibility info element**
0	0	0	0	1	0	0	0	eight octets follow this octet.
1	0	0	0	0	0	1	1	only window size and N1 was understood and analyzed.
0	1	0	1	0	1	1	1	WB4JFI-1 digipeater has checked for compatibility.
0	1	0	0	0	0	1	0	
0	0	1	1	0	1	0	0	
0	1	0	0	1	0	1	0	
0	1	0	0	0	1	1	0	
0	1	0	0	1	0	0	1	
0	1	1	0	0	0	1	1	
(frame check sequence)								

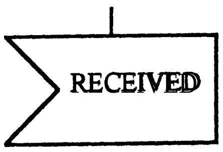
order of octet transmission

Figure 9 -- Example of an XID frame.

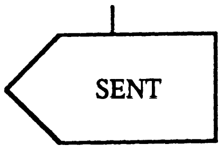
SOURCE/DESTINATION DATA LINK MANAGEMENT

Summary of Primitives, States, Queues, Flags, Parameters, Errors, and Timers

MDL Primitives

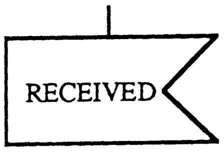


MDL Negotiate Request

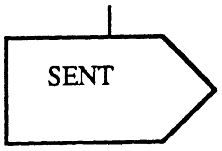


MDL Negotiate Confirm
MDL Error

LM Primitives



XID command
XID response



XID command
XID response

States

o-ready
1 -- negotiating
2 -- changeover

Queues

none.

Error Codes

A -- XID command received without P bit set to '1'.
B -- unexepected XID response received.
C -- NM201 retries attempted; no valid response received.
D -- XID response received without F bit set to '1'.
E -- changeover failed.

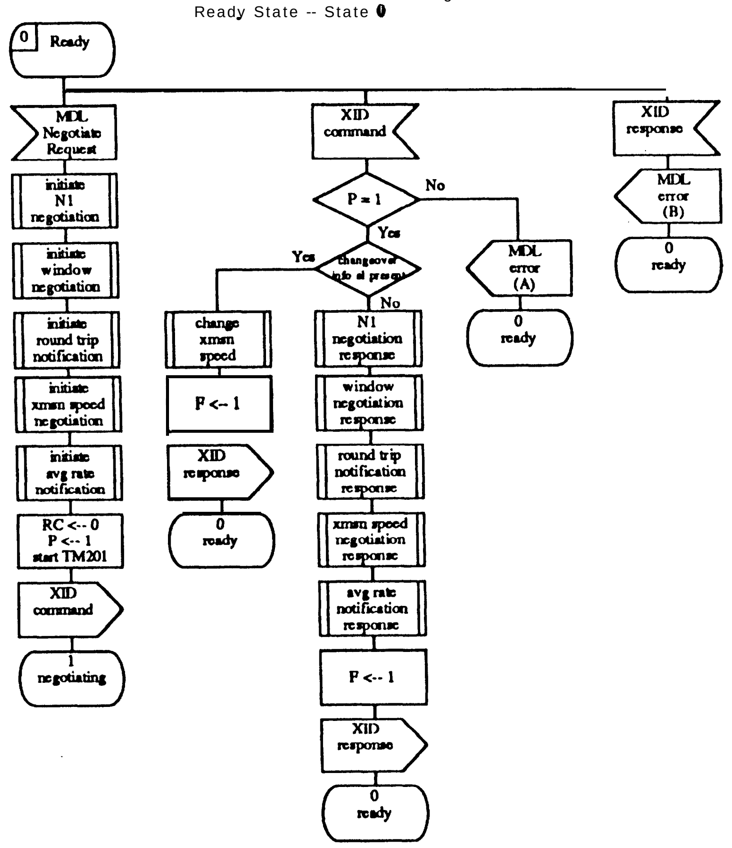
Flags & Parameters

changeover -- set when changeover procedure is to be executed next.
NM201 -- retry limit.
RC -- retry counter.

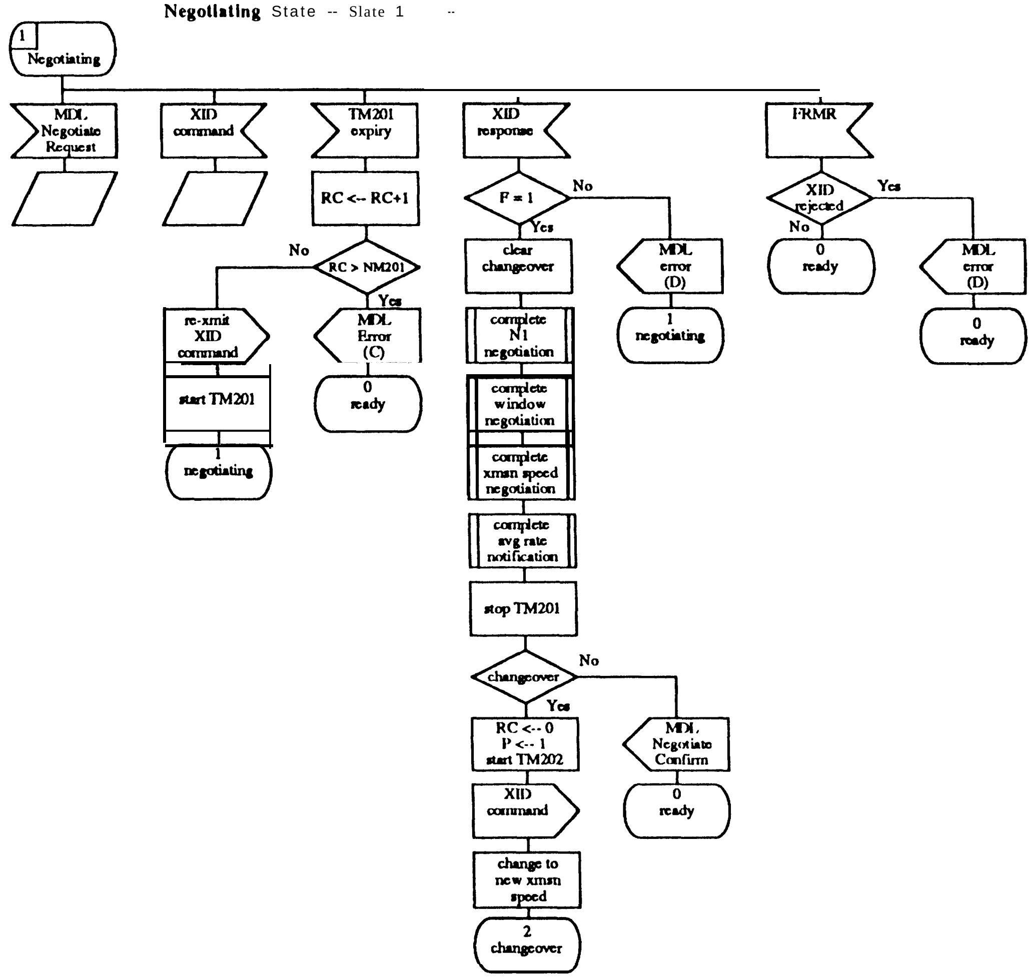
Timers

TM201 -- supervises exchange of XID frames for negotiation.
TM202 -- supervises exchange of XID frames for changeover.

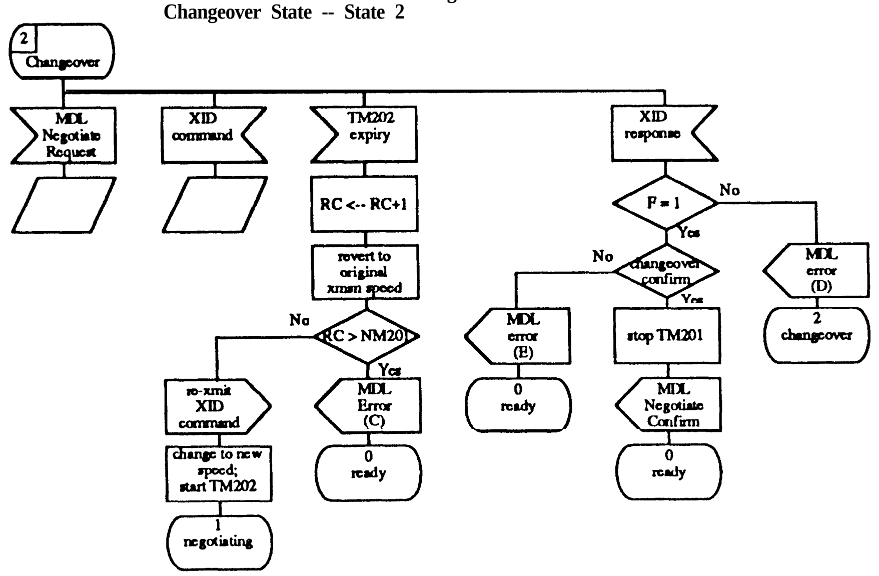
Source/Destination Station Data Link Manrgement
Ready State -- State 0



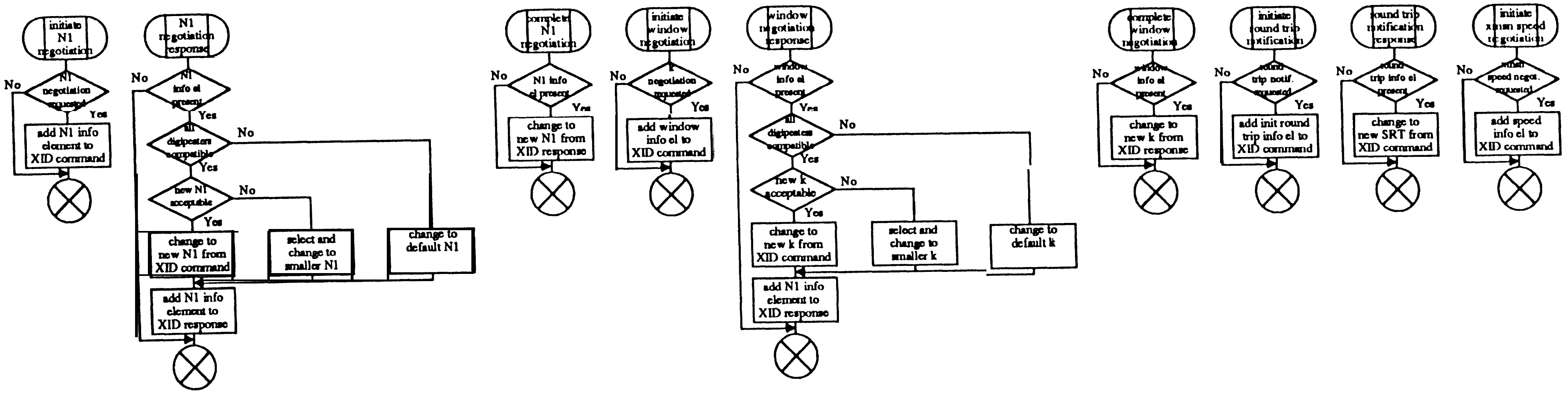
Source/Destination Station Data Link Management
Negotiating State -- Slate 1

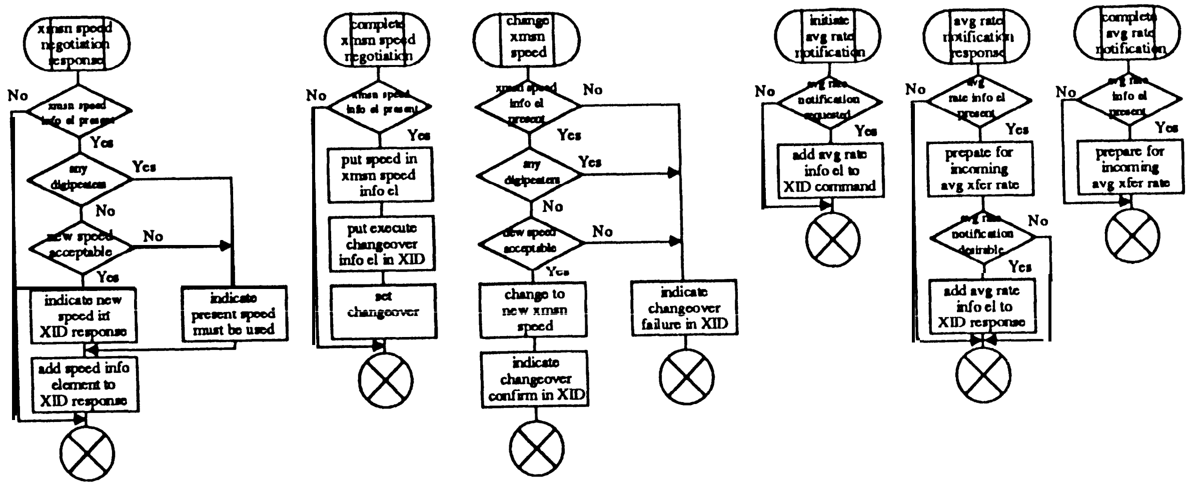


Source/Destination Station Data Link Management
Changeover State -- State 2



Source/Destination Station Data Link Management
Subroutine8

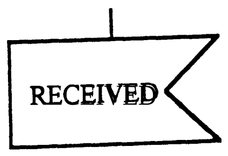

initiate N1 negotiation
N1 negotiation requested
No
Yes
add N1 info element to XID command
N1 negotiation response
N1 info el present
No
Yes
all digipeaters compatible
No
Yes
new N1 acceptable
No
Yes
change to new N1 from XID command
select and change to smaller N1
change to default N1
add N1 info element to XID response
complete N1 negotiation
N1 info el present
No
Yes
change to new N1 from XID response
initiate window negotiation
k negotiation requested
No
Yes
add window info el to XID command
window negotiation response
window info el present
No
Yes
all digipeaters compatible
No
Yes
new k acceptable
No
Yes
change to new k from XID command
select and change to smaller k
change to default k
add N1 info element to XID response
complete window negotiation
window info el present
No
Yes
change to new k from XID response
initiate round trip notification
round trip notif. requested
No
Yes
add init round trip info el to XID command
round trip notification response
round trip info el present
No
Yes
change to new SRT from XID command
initiate xmsn speed negotiation
xmsn speed negot. requested
No
Yes
add speed info el to XID command


xmsn speed negotiation response
xmsn speed info el present
No
Yes
any digipeaters
Yes
No
new speed acceptable
No
Yes
indicate new speed in XID response
indicate present speed must be used
add speed info element to XID response
complete xmsn speed negotiation
xmsn speed info el present
No
Yes
put speed in xmsn speed info el
put execute changeover info el in XID
set changeover
change xmsn speed
xmsn speed info el present
No
Yes
any digipeaters
Yes
No
new speed acceptable
No
Yes
change to new xmsn speed
indicate changeover confirm in XID
indicate changeover failure in XID
initiate avg rate notification
avg rate notification requested
No
Yes
add avg rate info el to XID command
avg rate notification response
avg rate info el present
No
Yes
prepate for incoming avg xfer rate
avg rate notification desirable
No
Yes
add avg rate info el to XID response
complete avg rate notification
avg rate info el present
No
Yes
prepare for incoming avg xfer rate

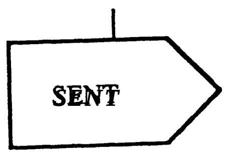
DIGIPEATER DATA LINK MANAGEMENT

Summary of Primitives, States, Queues, Flags, Parameters, Errors, and Timers

LM Primitives



XID command
XID response



XID command
XID response

Flags & Parameters

none.

States

0 -- ready

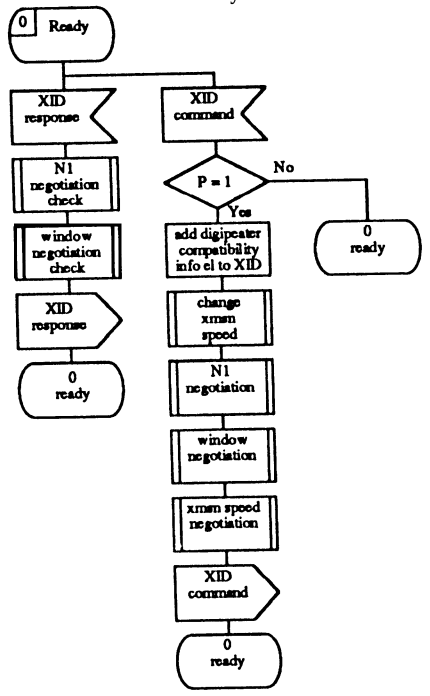
Timers

none.

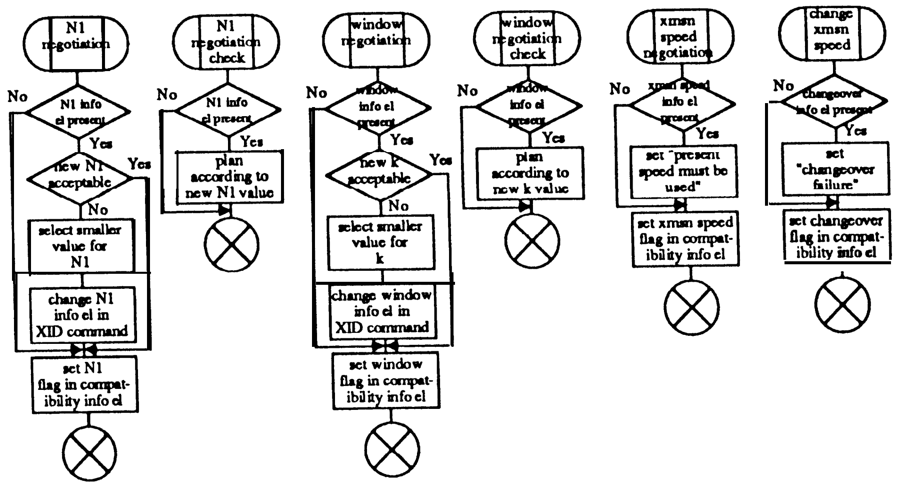
Queues

none.

Digipeater Station Data Link Management
Ready State -- State 0



Source/Destination Station Data Link Management
Subroutines



N1 negotiation
N1 info el present
No
Yes
new N1 acceptable
Yes
No
select smaller value for N1
change N1 info el in XID command
set N1 flag in compatibility info el
N1 negotiation check
N1 info el present
No
Yes
plan according to new N1 value
window negotiation
window info el present
No
Yes
new k acceptable
Yes
No
select smaller value for k
change window info el in XID command
set window flag in compatibility info el
window negotiation check
window info el present
No
Yes
plan according to new k value
xmsn speed negotiation
xmsn speed info el present
No
Yes
set "present speed must be used"
set xmsn speed flag in compatibility info el
change xmsn speed
changeover info el present
No
Yes
set "changeover failure"
set changeover flag in compatibility info el

Segmenter
State Machine

Eric L. Scace K3NA
10701 Five Forks Road
Frederick MD 21701
USA

0. Summary

This paper is part of a series of papers which provide extended finite state machine representations for AX.25 and related protocols. The state machines are depicted using state description language (SDL) graphic conventions from the Z.100 series of Recommendations developed by the International Telegraph and Telephone Consultative Committee (CCITT) of the International Telecommunications Union (ITU). An extended finite state machine representation of a communications protocol such as AX.25 avoids the ambiguities associated with prose descriptions. These descriptions also compell the protocol designer to confront many of the error scenarios which arise on a communications path, and simplify the implementor's task of producing correct solutions which will interwork with solutions created by others.

This particular paper describes an extended finite state machine to permit long units of data to be transmitted over an X.25 data link, even when that data link is restricted to smaller data units. The state machine has two parts:

-- a segmenter, which divides lengthy data units into a chain of segments for transmission; and,

-- a reassembler, which accumulates segments back together into the original long data unit.

For simplicity, these state machines are referred together as a "segmenter".

1. Status of Proposal

The SDL description here is a draft. The ARRL Digital Committee intends to include this machine as an Annex of the upcoming publication of AX.25 Revision 2.1.

AX.25 Revision 2.0 permits units of data up to 256 octets in length to be transmitted in a single AX.25 I or UI frame. From time to time, applications wish to exchange units of data which exceed this limit. Heretofore, there has been only one solution: to use a higher layer protocol to divide the units of data into smaller units for transmission. The drawback of this approach, particularly in a TCP/IP environment, is that the higher layer protocol headers must be inserted on each subdivided unit. This results in increased overhead on the link.

The situation degrades even *further* when links of poorer transmission quality, such as those experienced on HF radio, are encountered. On these poorer links, it is desirable to keep the AX.25 frames even shorter than normal. Higher layer protocol headers can easily consume more than 50% of a short AX.25 frame on such links.

This proposal remedies these limitations. The segmentor is a simple process which divides long data units into smaller segments for transmission, attaching a two octet header to each segment. At the receiving end, segments are reassembled into the original data unit. Overhead is kept to a minimum throughout, and steps are taken to prevent deadly embrace situations from arising in the buffer managements of both stations on the link.

The following material is still in a draft state. You are invited to review and comment on this material. Comments are desired so that the final publication is as useful as possible to its readers.

2. Features of the Segmentor SDL Machine

The simplex physical SDL machine includes the following features:

a) no overhead when segmentation is not required.

b) low overhead when segmentation is required.

c) provisions to prevent resource deadlocks (deadly embraces) on the data link during segmentation.

d) minimum delay impact during segmentation.

e) adaptability to any value of AX.25 data link N1 (maximum number of octets in the information field of an I or UI frame, excluding the flags, address, control, and frame check sequence fields, as well as 0-bits inserted for transparency).

f) maximum of 128 segments. For the standard AX.25 data link N1 value of 256 octets, this means that a single higher-layer protocol unit, such as an IP datagram, of 32k bytes can be transmitted under a single higher-layer protocol header.

g) protection to ensure that a loss of one or more segments is detected and reported to the AX.25 higher layer user.

h) useable on both connectionless (UI frame) and connection-oriented (I frame) AX.25 transfer modes; however, the use of I frames is *strongly* recommended to avoid aborting a segmented data unit.

3. Location in Overall Model

This SDL machine resides within the data link layer of the standard Open Systems Interconnection reference model. It interacts with the AX.25 user above *it,* and with the data link state machine below it. Neither the AX.25 user nor the data link SDL machine need be aware of the presence of the segmenter SDL machine.

In fact, to simplify the representation of the logic, there are two independent SDL machines: the segmenter and the reassembler.

3.1 Segmenter SDL Machine

The AX.25 user assumes that it is directing the operation of the data link SDL machine through the use of DL primitives described in an accompanying paper. However, when the segmenter SDL machine exists, it intercepts DL primitives and examines them. Only the following DL primitives will be candidates for modification by the segmenter SDL machine:

DL Data Request -- The AX.25 user employs this primitive to provide information to be transmitted using connection-oriented procedures.; i.e., using I frames The segmenter SDL machine examines the quantity of data to be transmitted. If the quantity of data to be transmitted exceeds the data link parameter N1, the segmenter SDL machine swings into action. The segmentation procedures ruthlessly chop up the data into segments of length N1-2 octets. Each segment is prepended with a two octet header. See Figures 1 and 2. The segments are then turned over to the data link SDL machine for transmission, using multiple DL Data Request primitives. All segments are turned over immediately; therefore the data link SDL machine will transmit them consecutively on the data link.

DL Unit Data Request -- The AX.25 user employs this primitive to provide information to be transmitted using connectionless procedures; i.e., using UI frames. The segmenter SDL machine examines the quantity of data to be transmitted. If the quantity of data to be transmitted exceeds the data link parameter N1, the segmenter SDL machine swings into action again, as described above. The segments are turned over to the data link SDL machine for transmission, using multiple DL Unit Data Request primitives. All segments are turned over immediately; therefore the data link SDL machine will transmit them consecutively on the data link.

DL Data Request and DL Unit Data Request primitives which are accompanied with a quantity of data less than N1 will pass transparently through the segmenter SDL machine. No segment header is prepended to the data.

3.2 Reassembler SDL Machine

The data link SDL machine delivers various primitives to the AX.25 user via the reassembler SDL machine. All primitives from the data link SDL machine are delivered transparently, without modification, exceed for the following:

DL Data Indication -- This primitive is examined by the reassembler SDL machine. If the accompanying received data begins with an octet encoded other than 0000 1000, it is assumed that this octet is a conventional PID (as presently described by AX.25 Revision 2.0). If the received data begins with an octet encoded 0000 1000, the reassembler SDL machine assumes that a segment header is present. After various checks for errors, this segment and all remaining segments received in subsequent DL Data Indication primitives are assembled together to recreate the original large data unit. Upon receipt and aggregation of the last segment, the entire large data unit is delivered to the AX.25 user via a single DL Data Indication primitive.

DL Unit Data Indication -- An identical process is followed for the DL Unit Data Indication primitive; the primitive is examined by the reassembler SDL machine. If the accompanying received data begins with an octet encoded other than 0000 1000, it is assumed that this octet is a conventional PID (as presently described by AX.25 Revision 2.0). If the received data begins with an octet encoded 0000 1000, the reassembler SDL machine assumes that a segment header is present. After various checks for errors, this segment and all remaining segments received in subsequent DL Unit Data Indication primitives are assembled together to recreate the original large data unit. Upon receipt and aggregation of the last segment, the entire large data unit is delivered to the AX.25 user via a single DL Unit Data Indication primitive.

DL Data Indication and DL Unit Data Indication primitives containing a conventional PID will pass transparently through the reassembler SDL machine and be delivered immediately to the AX.25 user.

3.3 Summary

Under normal operation, therefore, the net result is that the sending AX.25 user provides a single DL Data Request primitive, containing a large unit of data, to the overall set of SDL machines. Segmentation occurs in the segmenter SDL machine; the data link, link multiplexor, and physical SDL machines work together to transmit the segments across the link to the receiving station. At the receiving station, the physical, link multiplexor, and data link SDL machines work together to receive the incoming segments; reassembly occurs in the reassembler SDL machine. The receiving AX.25 user then receives a single DL Data Request primitive containing the original large unit of data.

The entire set of SDL machines works together to transparently move large data units across the data link.

If an error in reassembly is detected, that error is reported to the AX.25 user with a DL Error Indication primitive.

4. Internal Operation of the SDL Machines

The internal states, error codes, and timers are summarized on the first page of the SDL diagram.

4.1 Internal Operation of the Segmenter SDL Machine

The segmenter SDL machine operation is quite straightforward. Only one state exists for this machine.

4.2 Internal Operation of the Reassembler SDL Machine

The reassembler SDL machine resides in the Null state until the start of a segmented data stream is detected. At this point, a check is made to ensure that the first segment received is, in fact, the first segment of the message. This check is performed by examining octet 2, bit 8 of the segment header (see Figure 2). If *this* is *not* the first segment, then the reassembler SDL machine assumes that the actual first segment was lost somewheres, and signals an error. All segments will be discarded as they are received.

Assume now that the first segment was received correctly. The reassembler SDL machine then allocates sufficient storage to receive all the remaining segments; this prevents deadly embrace (resource deadlock) conditions. The reassembler SDL machine enters either the reassembling data state (if segments are arriving in I frames) or the reassembling unit data state (if segments are arriving in UI frames). A lengthy timer supervises both of these states; its purpose is to protect the reassembly process from hanging if a very long delay happens to occur (e.g., the remote station breaks down and never completes transmission). This timer is designated TR210: "R" for reassembler; "2" for level 2, the data link level of the OSI open systems interconnection protocol architecture; and "10" simply to avoid confusion with any other timers in this family of SDL machines.

Each incoming segment is examined to ensure that it is indeed the next expected segment. If the loss of a segment is detected, the entire accumulation of data is discarded and an error notification is provided to the AX.25 user. No attempt is made by the segmenter and reassembler SDL machines to recover segmented data units; this is left to the higher level AX.25 user. Rather, the reassembler SDL machine works to ensure that large data units are completely received and correctly reassembled over the data link. In other words, segmentation error detection is provided, but no segmentation error correction is provided.

The reassembler SDL machine also insists that, once the transmission of a segmented large data unit is begun, all segments will be transmitted until the complete large data unit has been transferred. No other event is permitted to occur over the data link. This constraint is imposed for two reasons:

- to ensure that stations with multiple data links minimize the amount of buffer capacity tied up in partially received or transmitted large data units. This in turns reduces the likelihood of link busy conditions (RNR) on connection-oriented links and of discard on connectionless links; and,
- to minimize the delay in transmission of large data units, once large data unit has reached the top of the queue. This in turn means that the AX.25 users, having sent or received the large wad of information, can then move on with the job of processing it and satisfy the thirst of their human users (local or remote) for knowledge.

5.

As mentioned § 2 (h) above, the use of connection-oriented data link procedures is recommended when segmentation is used across data links with even moderately low collision levels. If connectionless data link procedures (UI frames) are used to carry segments, the loss of a single UI frame will result in the loss of the entire segmented large data unit; higher level attempts at recovery will increase the amount of congestion on the physical channel.

The simple, CCITT-standardized segmentation procedure here provide two important advancements in packet radio capabilities:

- the ability to carry large data units across a data link with a minimum of overhead (less than 1% for the standard N1 value of 256 octets).
- the ability to reduce N1 for error-prone data links (such as those *over* HF radio channels) to improve the net successful information transfer rate without having to replicate lengthy higher level protocol headers.

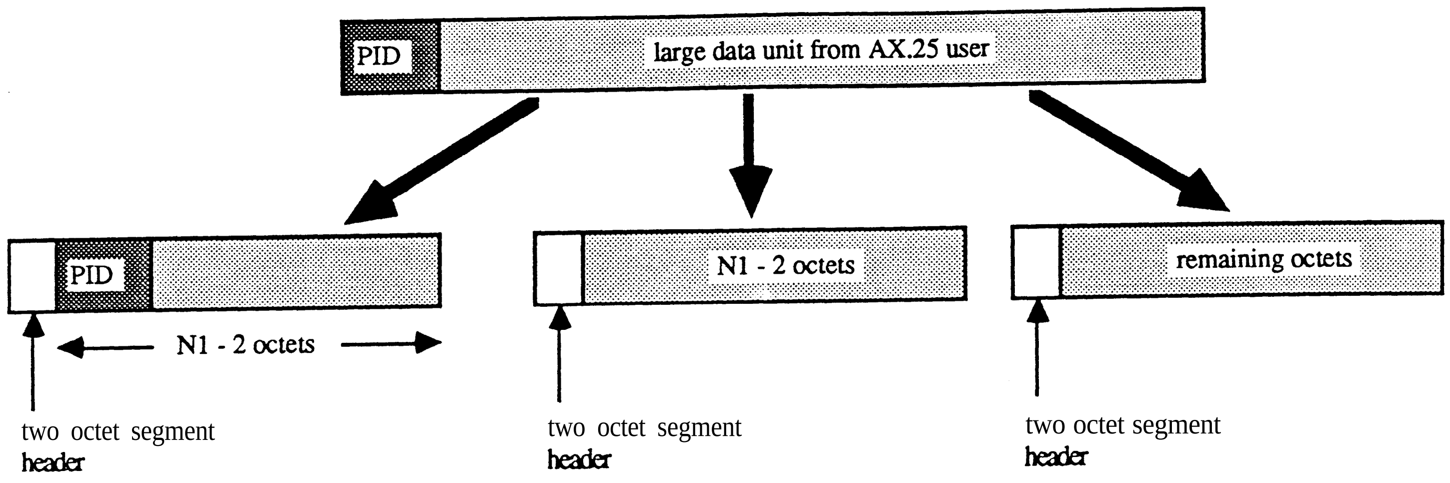


Figure 1 -- Segmentation Process

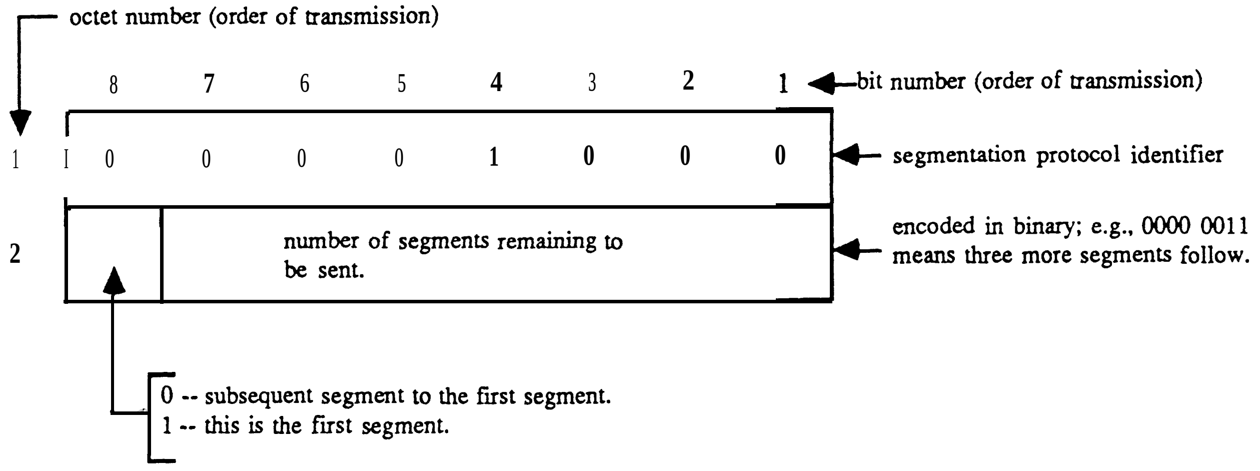
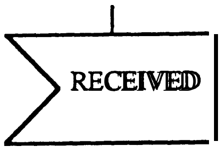

octet number (order of transmission)
8 7 6 5 4 3 2 1
bit number (order of transmission)
1
0 0 0 0 1 0 0 0
segmentation protocol identifier
2
number of segments remaining to be sent.
encoded in binary; e.g., 0000 0011 means three more segments follow.
0 -- subsequent segment to the first segment.
1 -- this is the first segment.

Figure 2 -- Segment Header Format

SEGMENTER

Summary of Primitives, States, Queues, Flags, Parameters, Errors, and Timers

DL Primitives



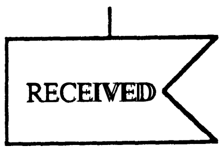
DL Data Request
DL Unit Data Requent
Note -- other DL primitives are passed transparently.

SENT

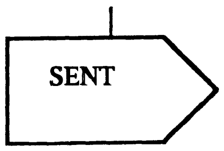
c

DL Data Indication
DL Unit Data Indication
DL Error Indication
Note -- other DL primitives are passed transparently.

DL Primitives



DL Data Indication
DL Unit Data Indication
Note -- all other DL primitives are passed transparently.



DL Data Request
DL Unit Data Request
Note -- all other DL primitives are passed transparently.

Segmenter States

0 -- ready

Reassembler States

0 -- null
1 -- reassembling data
2 -- reassembling unit data

Queues

none.

Error Codes

Z -- reassembly error.

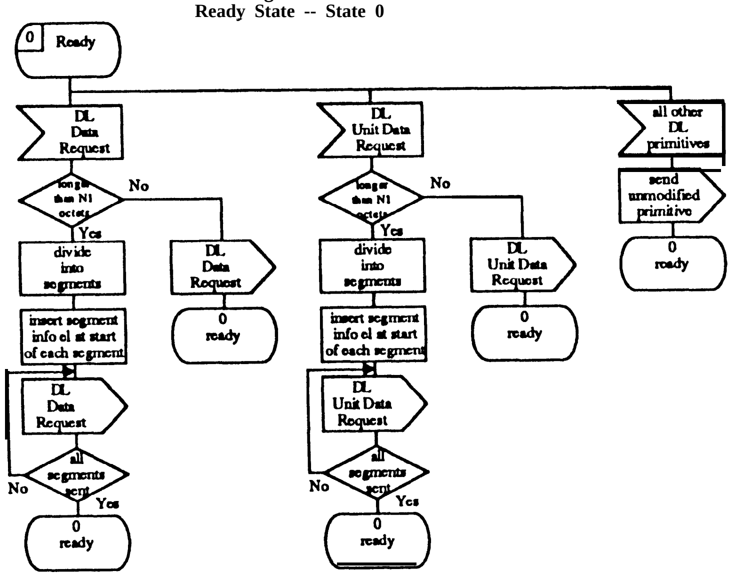
Flags & Parameters

N -- number of segments remaining to be reassembled.

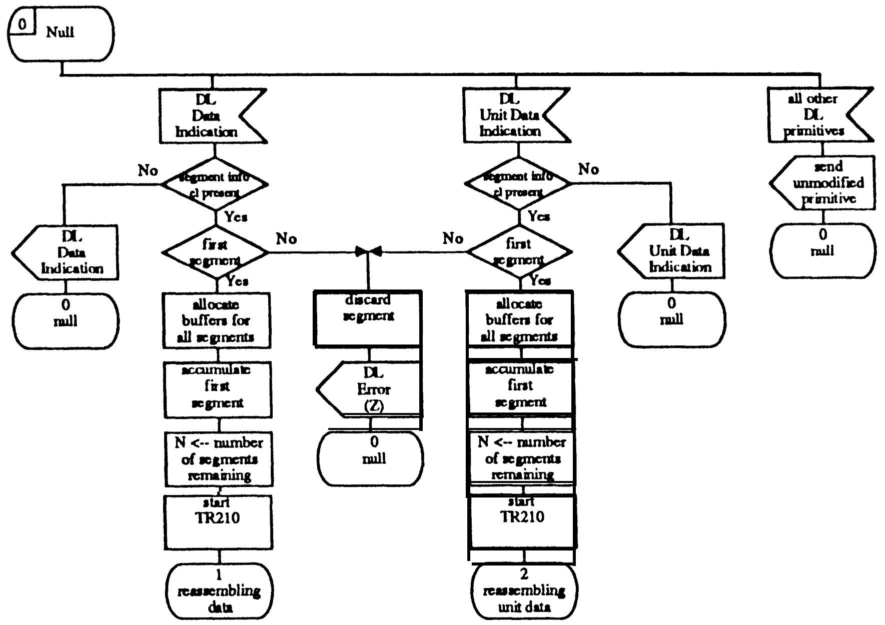
Timers

TR210 -- time limit for receipt of next segment.

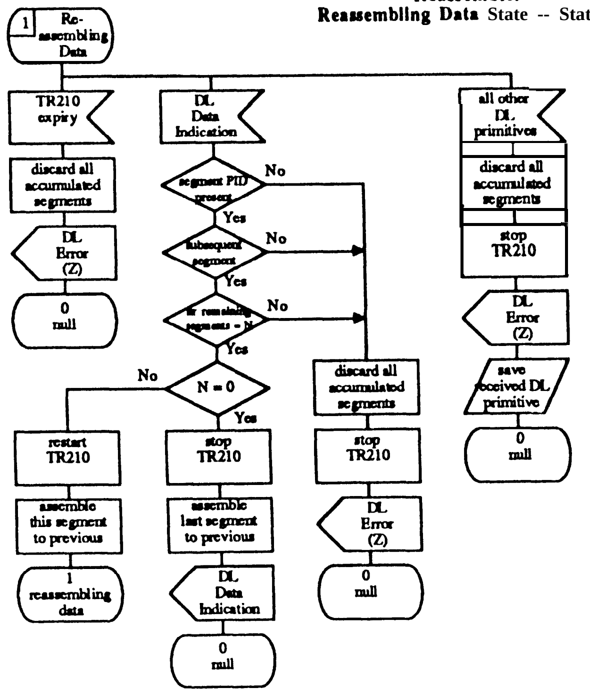
Segmcnter
Ready State -- State 0



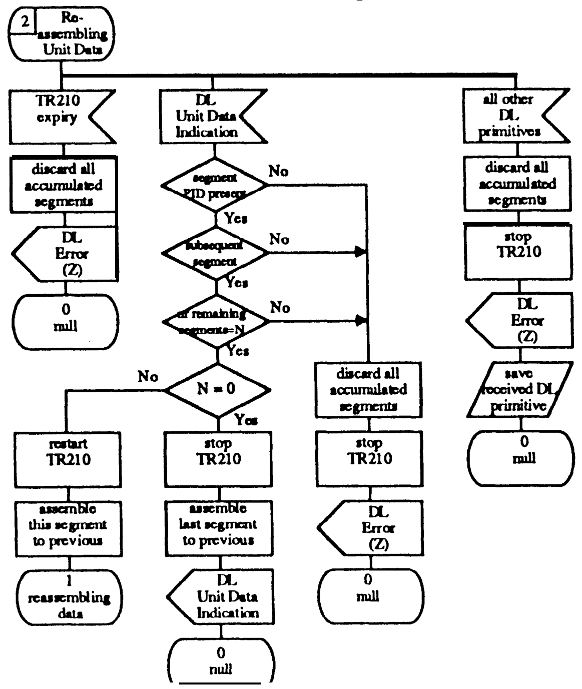
Rtrrrembler
Null State -- Slate 0



Reassembler
Reassembling Data State -- State 1



Reassembler
Reassembling Unit Data State -- State 2


2 Re-assembling Unit Data
TR210 expiry
discard all accumulated segments
DL Error (Z)
0 null
DL Unit Data Indication
segment PID present
No
Yes
subsequent segment
No
Yes
nr remaining segments=N
No
Yes
N = 0
No
Yes
restart TR210
assemble this segment to previous
1 reassembling data
stop TR210
assemble last segment to previous
DL Unit Data Indication
0 null
discard all accumulated segments
stop TR210
DL Error (Z)
0 null
all other DL primitives
discard all accumulated segments
stop TR210
DL Error (Z)
save received DL primitive
0 null